«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Вычислительная математика»

Отчет

По лабораторной работе №5

Вариант 1

Студент:

Алхимовици А.

P3210

Преподаватель:

Наумова Н. А.

Санкт-Петербург, 2025 г.

Цель лабораторной работы:

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

1 Вычислительная реализация задачи:

1. Выбрать таблицу $y = f(x)$:

	X	y	вариант	X1	X2
Таблица 1.1	0.25	1.2557	1	0.251	0.402
	0.30	2.1764			
	0.35	3.1218			
	0.40	4.0482			
	0.45	5.9875			
	0.50	6.9195			
	0.55	7.8359			

2. Построить таблицу конечных разностей:

№	x	y	Δ1	Δ2	Δ3	Δ4	Δ5	Δ6
0	0.25	1.2557	0.9207	0.0247	-0.0437	1.0756	-4.1277	10.1917
1	0.30	2.1764	0.9454	-0.0190	1.0319	-3.0521	6.0640	
2	0.35	3.1218	0.9264	1.0129	-2.0202	3.0119		
3	0.40	4.0482	1.9393	-1.0073	0.9917			
4	0.45	5.9875	0.9320	-0.0156				
5	0.50	6.9195	0.9164					
6	0.55	7.8359						

3. Вычислить значения функции для аргумента $X1$ используя интерполяционную формулу Ньютона

Воспользуемся формулой Ньютона для интерполирования вперед(первая формула), так как X1 = 0.251 лежит в левой половине отрезка

Для $X_1 = 0.251$: $t = \frac{(x-x_0)}{h} = \frac{(0.251-0.25)}{0.05} = 0.02$

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

$$y(0.251) \approx 1.2557 + 0.02 * 0.9207 + \frac{0.02 \cdot (-0.98)}{2} * 0.0247 + \frac{0.02(-0.98)(-1.98)}{6} * (-0.0437) + \frac{0.02(-0.98)(-1.98)(-2.98)}{24} * (1.0756) + \frac{0.02(-0.98)(-1.98)(-2.98)(-3.98)}{120} * (-4.1277) + \frac{0.02(-0.98)(-1.98)(-2.98)(-3.98)(-4.98)}{720} * (10.1917) \approx 1.2476$$

$y(0.251) \approx 1.2476$

4. **Вычислить значения функции для аргумента X_2**, используя интерполяционную формулу **Гаусса**:
Воспользуемся первой формулой Гауса, так как X2 = 0.402 > a = 0.4

$$t = \frac{(x - x_0)}{h} = \frac{(0.402 - 0.4)}{0.05} = 0.04$$

№	x	y	Δ1	Δ2	Δ3	Δ4	Δ5	Δ6
-3	0.25	1.2557	0.9207	0.0247	-0.0437	1.0756	-4.1277	10.1917
-2	0.30	2.1764	0.9454	-0.0190	1.0319	-3.0521	6.0640	
-1	0.35	3.1218	0.9264	1.0129	-2.0202	3.0119		
0	0.40	4.0482	1.9393	-1.0073	0.9917			
1	0.45	5.9875	0.9320	-0.0156				
2	0.50	6.9195	0.9164					
3	0.55	7.8359						

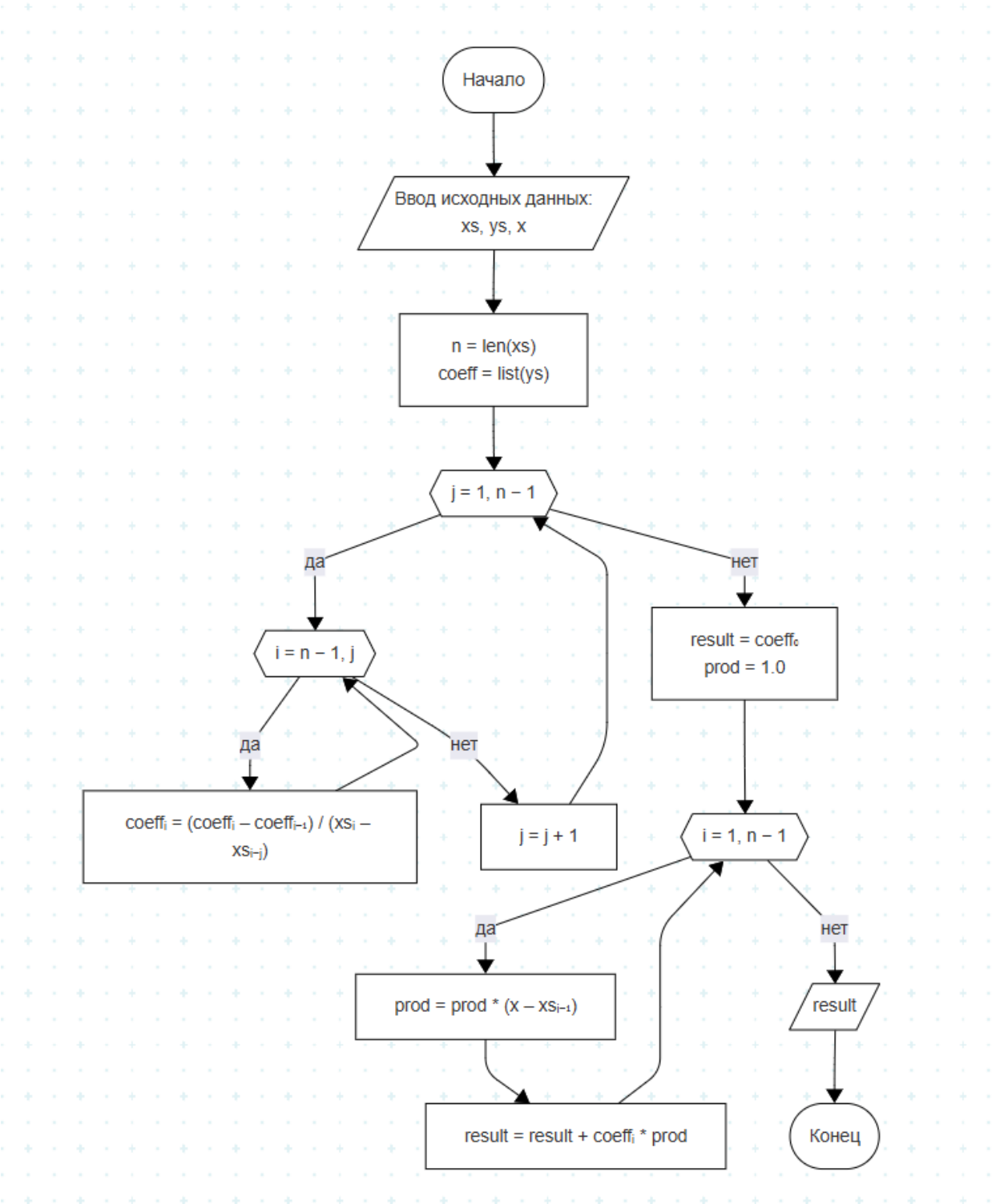
$$P_6(x) = y_0 + t\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_{-1} + \frac{(t+1)t(t-1)}{3!}\Delta^3 y_{-1} + \frac{(t-2)(t+1)t(t-1)}{4!}\Delta^4 y_{-2} + \frac{(t+2)(t+1)t(t-1)(t-2)}{5!}\Delta^5 y_{-2} + \frac{(t-3)(t+2)(t+1)t(t-1)(t-2)}{6!}\Delta^6 y_{-3}$$

$P_6(0.402)$=4.0482+0.04*(1.9393)+1.0129*0.04*(-0.96)/2-
2.0202*0.04*0.96*1.004/6-3.0521*1.96*0.96*1.04*0.04/24
+6.0640*0.04*0.96*1.004*1.96*2.004/120 -
10.1917*0.04*0.96*1.004*1.96*2.004*2.96/720≈4.084

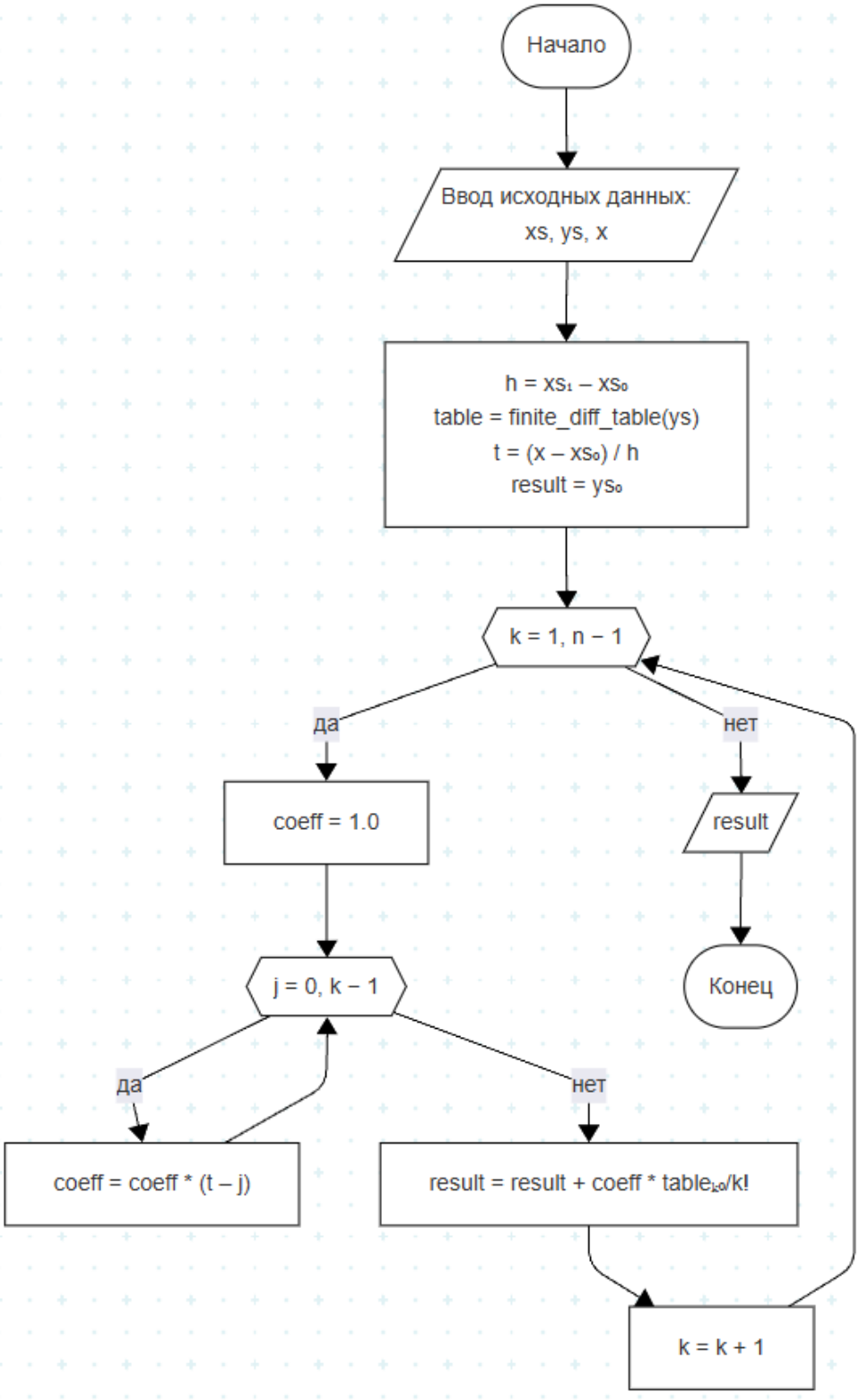
2. Программная реализация

Блок схемы

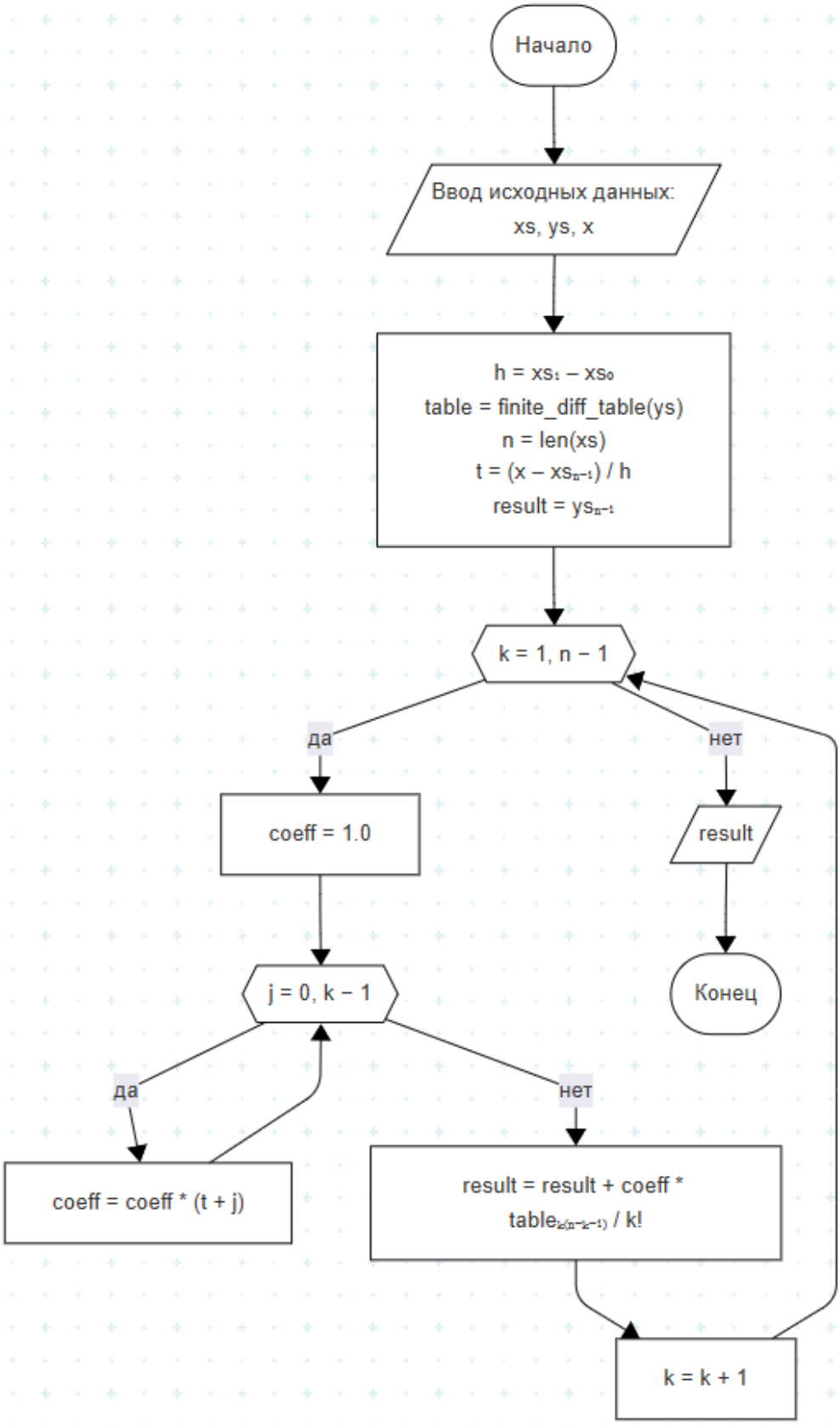
Многочлен Ньютона с разделенными разностями



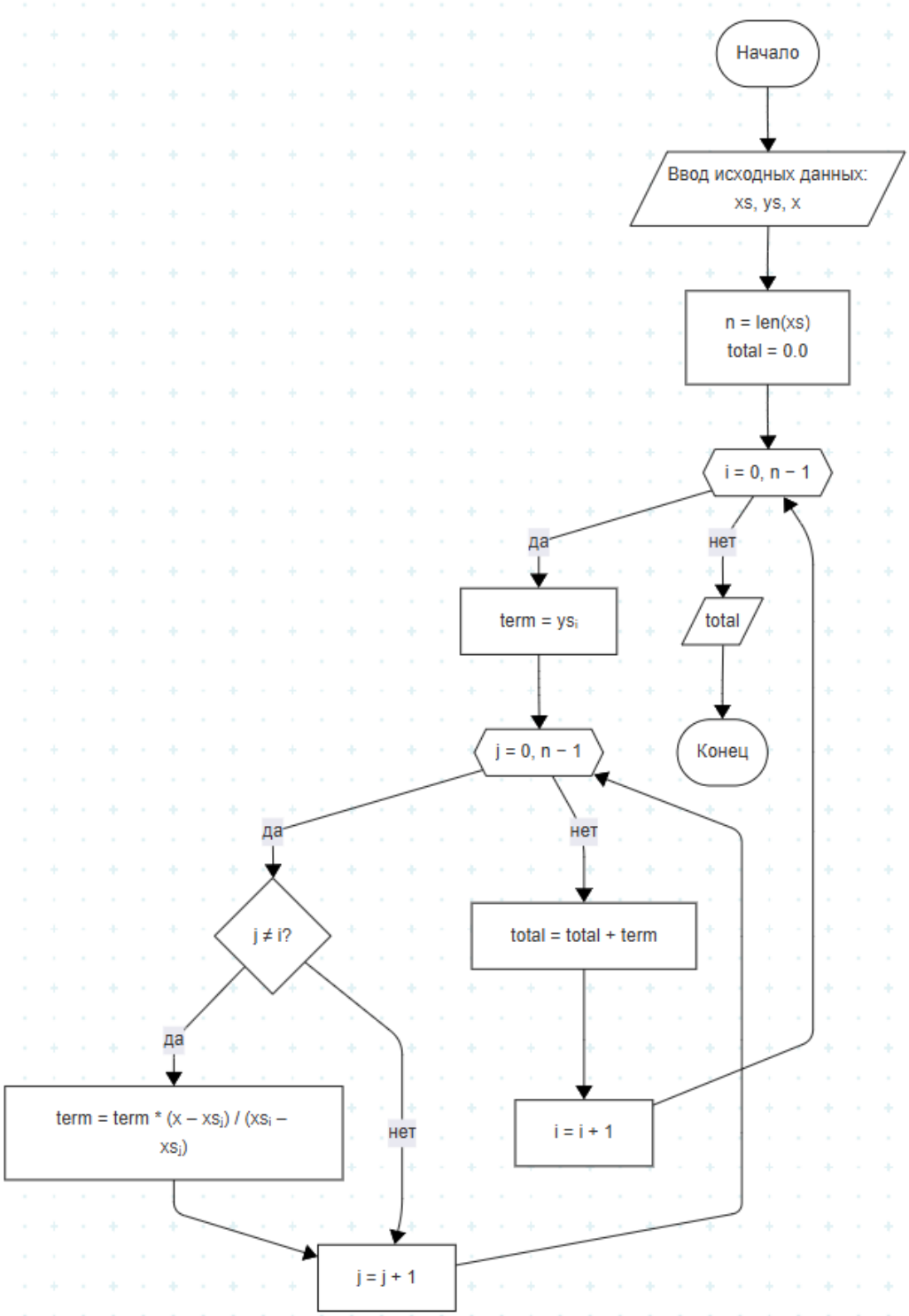
Формула прямых конечных разностей



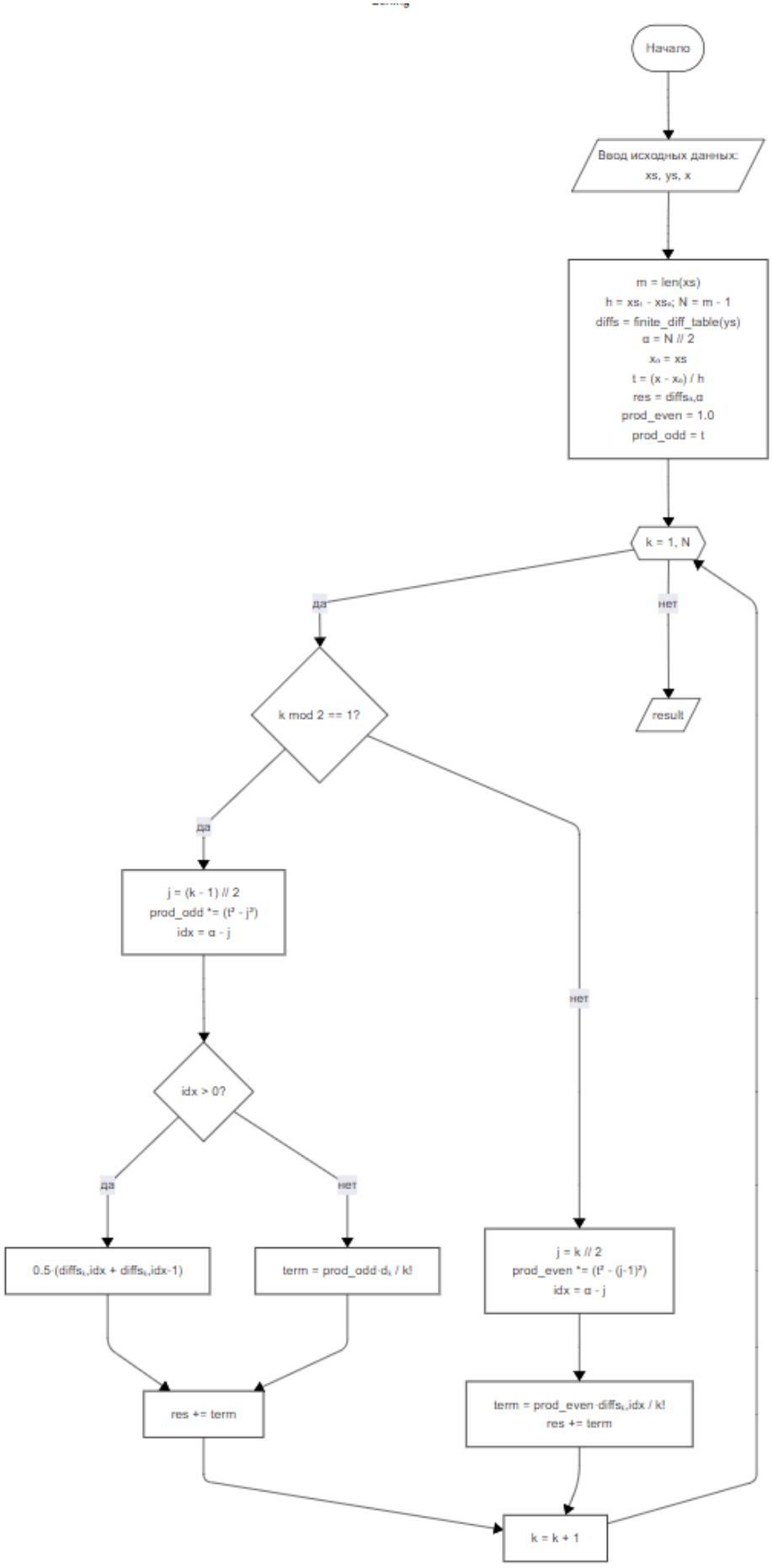
Формула обратных конечных разностей



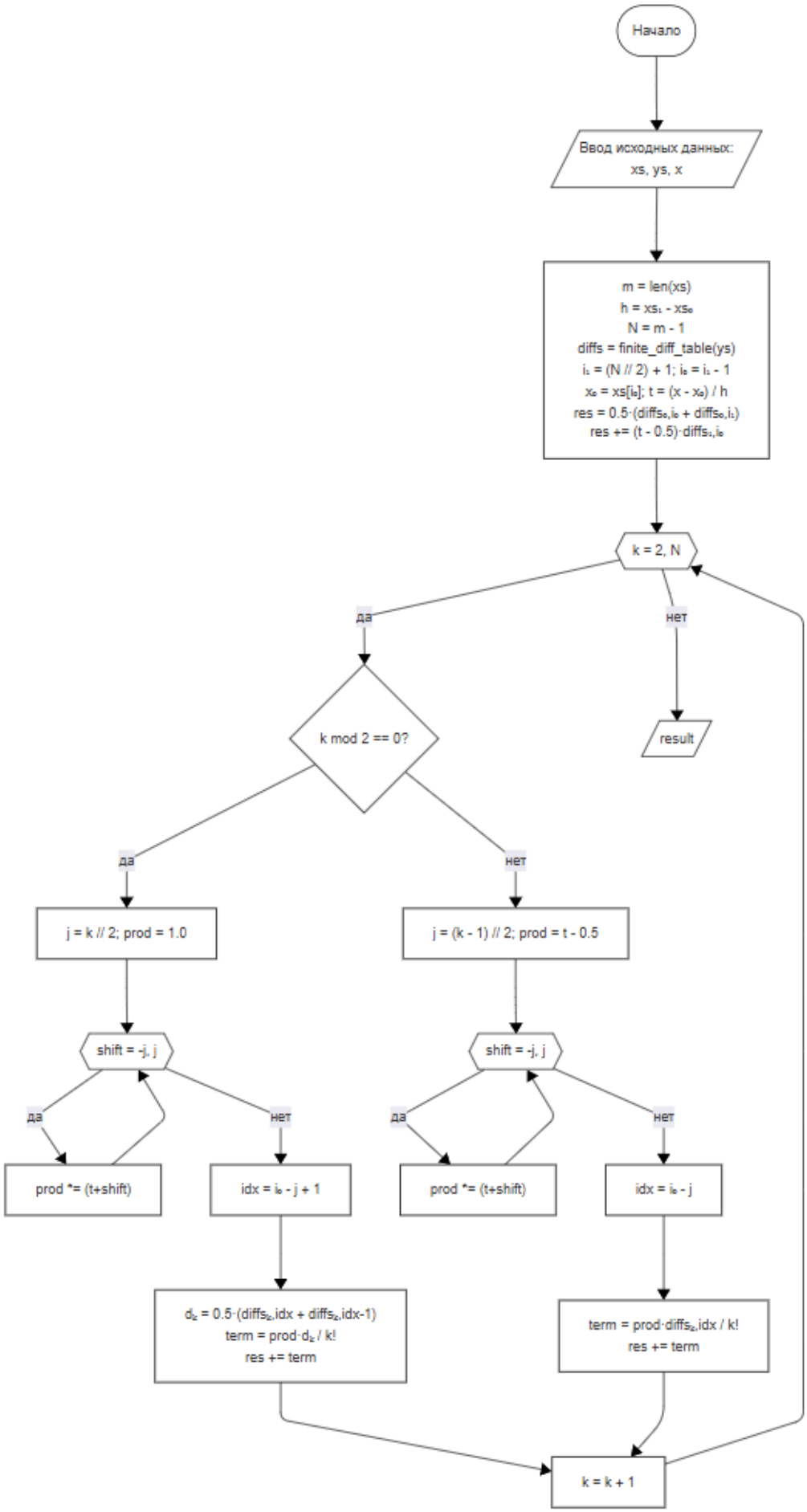
Интерполяция Лагранжа



Стирлинг:


Начало
Ввод исходных данных:
xs, ys, x
m = len(xs)
h = xs₁ - xs₀; N = m - 1
diffs = finite_diff_table(ys)
α = N // 2
x₀ = xs
t = (x - x₀) / h
res = diffs₀,α
prod_even = 1.0
prod_odd = t
k = 1, N
да
нет
result
k mod 2 == 1?
да
нет
j = (k - 1) // 2
prod_odd *= (t² - j²)
idx = α - j
idx > 0?
да
нет
0.5·(diffsₖ,idx + diffsₖ,idx-1)
term = prod_odd·dₖ / k!
j = k // 2
prod_even *= (t² - (j-1)²)
idx = α - j
term = prod_even·diffsₖ,idx / k!
res += term
res += term
k = k + 1

Бессел:


Начало
Ввод исходных данных:
xs, ys, x
m = len(xs)
h = xs₁ - xs₀
N = m - 1
diffs = finite_diff_table(ys)
i₁ = (N // 2) + 1; i₀ = i₁ - 1
x₀ = xs[i₀]; t = (x - x₀) / h
res = 0.5·(diffs₀,i₀ + diffs₀,i₁)
res += (t - 0.5)·diffs₁,i₀
k = 2, N
да
нет
result
k mod 2 == 0?
да
нет
j = k // 2; prod = 1.0
j = (k - 1) // 2; prod = t - 0.5
shift = -j, j
shift = -j, j
да
нет
да
нет
prod *= (t+shift)
idx = i₀ - j + 1
prod *= (t+shift)
idx = i₀ - j
d_k = 0.5·(diffs_k,idx + diffs_k,idx-1)
term = prod·d_k / k!
res += term
term = prod·diffs_k,idx / k!
res += term
k = k + 1

Листинг программы

https://github.com/senya-2011/Vu4Math/tree/main/lab5

```
def lagrange(xs, ys, x):
    """
    Интерполяция Лагранжа.
    """
    n = len(xs)
    total = 0.0
    for i in range(n):
        term = ys[i]
        for j in range(n):
            if j != i:
                term *= (x - xs[j]) / (xs[i] - xs[j])
        total += term
    return total

# Многочлен Ньютона с разделёнными разностями
def newton_divided(xs, ys, x):
    """
    Интерполяция Ньютона (разделённые разности).
    """
    n = len(xs)
    coeff = list(ys)
    for j in range(1, n):
        for i in range(n - 1, j - 1, -1):
            coeff[i] = (coeff[i] - coeff[i - 1]) / (xs[i] - xs[i - j])
    result = coeff[0]
    prod = 1.0
    for i in range(1, n):
        prod *= (x - xs[i - 1])
        result += coeff[i] * prod
    return result

# Многочлен Ньютона с конечными разностями (прямая формула)
def newton_forward(xs, ys, x):
    """Формула прямых конечных разностей. Только для равноотстоящих
узлов."""
    h = xs[1] - xs[0]
    table = finite_diff_table(ys)
    t = (x - xs[0]) / h
    result = ys[0]
    for k in range(1, len(xs)):
        coeff = 1.0
        for j in range(k):
            coeff *= (t - j)
        result += coeff * table[k][0] / factorial(k)
    return result

# Многочлен Ньютона с конечными разностями (обратная формула)
def newton_backward(xs, ys, x):
    """Формула обратных конечных разностей. Только для равноотстоящих
узлов."""
    h = xs[1] - xs[0]
    table = finite_diff_table(ys)
    n = len(xs)
    t = (x - xs[-1]) / h
    result = ys[-1]
    for k in range(1, n):
        coeff = 1.0
        for j in range(k):
            coeff *= (t + j)
        result += coeff * table[k][n - k - 1] / factorial(k)
    return result
```

```
def stirling(xs, ys, x):
    m = len(xs)

    h = xs[1] - xs[0]
    N = m - 1

    diffs = finite_diff_table(ys)

    alpha = N // 2
    x0 = xs[alpha]
    t = (x - x0) / h

    res = diffs[0][alpha]

    prod_even = 1.0
    prod_odd = t

    for k in range(1, N + 1):
        if k % 2 == 1:
            j = (k - 1) // 2
            if j > 0:
                prod_odd *= (t*t - j*j)
            idx = alpha - j
            if idx > 0:
                d_center = 0.5 * (diffs[k][idx] + diffs[k][idx-1])
            else:
                d_center = diffs[k][idx]
            term = (prod_odd * d_center) / factorial(k)
        else:
            j = k // 2
            if j > 0:
                prod_even *= (t*t - (j-1)*(j-1))
            idx = alpha - j
            term = (prod_even * diffs[k][idx]) / factorial(k)

        res += term

    return res


def bessel(xs, ys, x):
    m = len(xs)

    h = xs[1] - xs[0]
    N = m - 1

    diffs = finite_diff_table(ys)

    i1 = (N // 2) + 1
    i0 = i1 - 1

    x0 = xs[i0]
    t = (x - x0) / h

    res = 0.5 * (diffs[0][i0] + diffs[0][i1])

    res += (t - 0.5) * diffs[1][i0]

    for k in range(2, N + 1):
        if k % 2 == 0:
            j = k // 2
            prod = 1.0
            for shift in range(-j, j):
                prod *= (t + shift)
            idx = i0 - j + 1
            diff_avg = 0.5 * (diffs[k][idx] + diffs[k][idx-1])
```

```
            term = prod * diff_avg / factorial(k)
        else:
            j = (k - 1) // 2
            prod = (t - 0.5)
            for shift in range(-j, j):
                prod *= (t + shift)
            idx = i0 - j
            diff_val = diffs[k][idx]
            term = prod * diff_val / factorial(k)

        res += term

    return res
```

Примеры и результаты работы программы

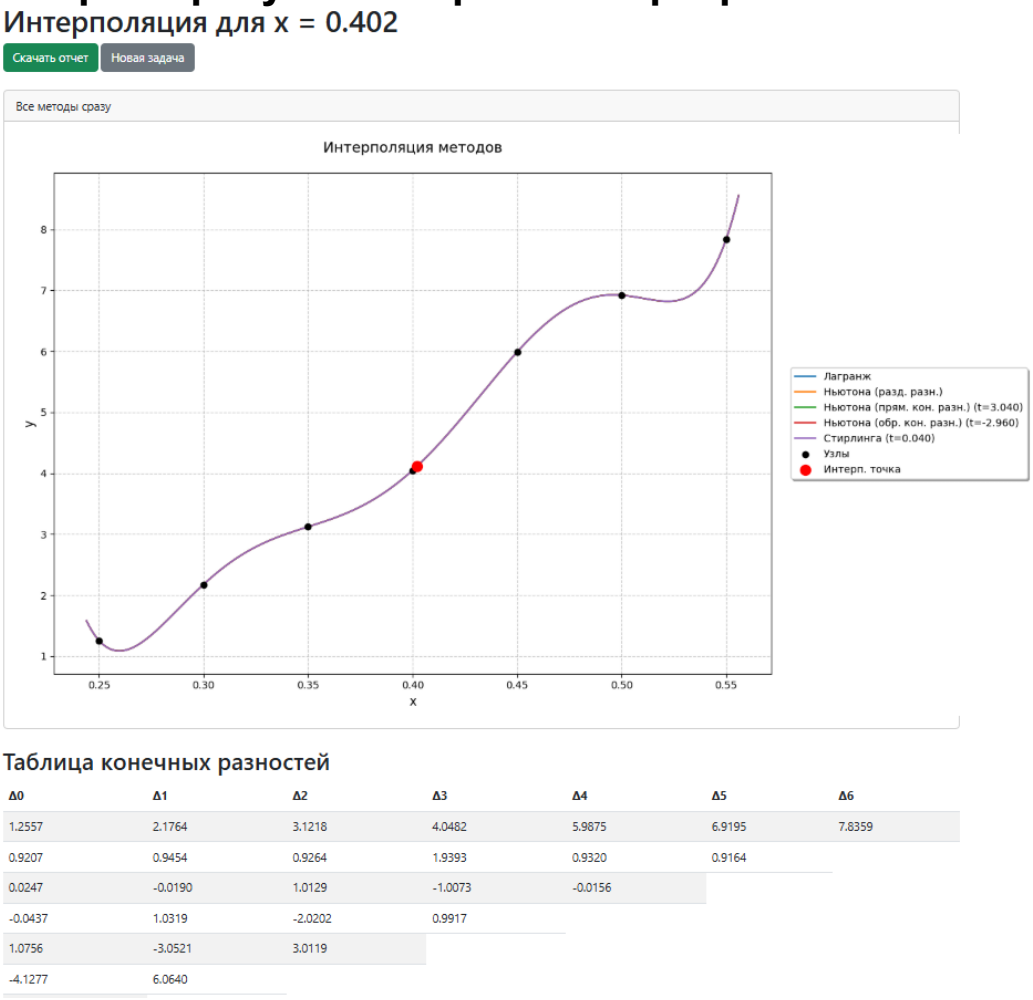


Таблица конечных разностей

Δ0	Δ1	Δ2	Δ3	Δ4	Δ5	Δ6
1.2557	2.1764	3.1218	4.0482	5.9875	6.9195	7.8359
0.9207	0.9454	0.9264	1.9393	0.9320	0.9164	
0.0247	-0.0190	1.0129	-1.0073	-0.0156		
-0.0437	1.0319	-2.0202	0.9917			
1.0756	-3.0521	3.0119				
-4.1277	6.0640					

Вывод:

В ходе выполнения лабораторной работы я изучил и практично применил интерполяционные методы Ньютона и Гаусса для обработки табличных данных. Эти методы позволили аппроксимировать значения функции в точках, отсутствующих в исходном наборе.

Разработанная программа обеспечила вычисление значений функции в заданных точках с использованием обоих методов. Проведённый сравнительный анализ показал совпадение

результатов, что свидетельствует о правильной реализации и работоспособности выбранных алгоритмов.