Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа **№5**
«Интерполяция функции»

по дисциплине «Вычислительная математика»

Вариант: **13**

Преподаватель:
Наумова Надежда Александровна

Выполнил:
Саранча Павел Александрович
Группа: P3209

Санкт-Петербург, 2025 г.

Цель 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	N варианта	X_1	X_2
Таблица 1.3	1.10	0.2234	**13**	1.168	1.463
	1.25	1.2438			
	1.40	2.2644			
	1.55	3.2984			
	1.70	4.3222			
	1.85	5.3516			
	2.00	6.3867			

2. Построить таблицу конечных разностей:

№	x_i	y_i	Δy_i	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^4 y_i$	$\Delta^5 y_i$	$\Delta^6 y_i$
0.	1,1	0,2234	1,0204	0,0002	0,0132	-0,0368	0,0762	-0,1313
1.	1,25	1,2438	1,0206	0,0134	-0,0236	0,0394	-0,0551	
2.	1,4	2,2644	1,034	-0,0102	0,0158	-0,0157		
3.	1,55	3,2984	1,0238	0,0056	0,0001			
4.	1,7	4,3222	1,0294	0,0057				
5.	1,85	5,3516	1,0351					
6.	2	6,3867						

3. Вычислить значения функции для аргумента X_1, используя первую или вторую интерполяционную формулу **Ньютона**:

Воспользуемся формулой Ньютона для интерполирования **вперед**, так как $X_1 = 1.168$ лежит в левой половине отрезка.

Для $X_1 = 1.168$: $t = \frac{(x - x_n)}{h} = \frac{(1.168 - 1.1)}{0.15} = 0.453$

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

$$y(1.168) \approx 0.2234 + 0.453 * 1.0204 + \frac{0.453(0.453-1)}{2} * 0.0002 + \frac{0.453(0.453-1)(0.453-2)}{6} * 0.0132 + \frac{0.453(0.453-1)(0.453-2)(0.453-3)}{24} * (-0.0368\,) + \frac{0.453(0.453-1)(0.453-2)(0.453-3)(0.453-4)}{120} * 0.0762 + \frac{0.453(0.453-1)(0.453-2)(0.453-3)(0.453-4)(0.453-5)}{720} * (-0.1313)$$

$$y(1.168) \approx 0.69337$$

4. Вычислить значения функции для аргумента *X*₂, используя первую или вторую интерполяционную формулу **Гаусса**:

Центральная точка $a = 0.65$, $X_2 = 0.645 < 0.65$, то есть $x < a$ → используем **вторую** интерполяционную формулу Гаусса.

$$\text{t} = \frac{(x - x_0)}{h} = \frac{(1.463 - 1.55)}{0.15} = -0.58$$

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

$$y(1.463) \approx 3.2984 + (-0.58) * 1.034 + \frac{-0.58(0.42)}{2} * (-0.0102) + \frac{(-0.58)(0.42)(-1.58)}{6} * (-0.0236) + \frac{(-0.58)(0.42)(-1.58)(1.42)}{24} * (0.0394) + \frac{(-0.58)(0.42)(1.42)(-2.58)}{120} * (0.0762) + \frac{(-0.58)(0.42)(-1.58)(1.42)(-2.58)(2.42)}{720} * (-0.1313)$$

$$y(1.463) \approx 2.69903$$

2. Программная реализация задачи

https://github.com/PaulLocust/comp_math_lab5

Листинг методов:

```python
def lagrange_polynomial(xs, ys, n, x):
    """
    Интерполяция по методу Лагранжа.

    Входные параметры:
    xs : массив координат узлов интерполяции
    ys : массив значений функции в узлах
    n  : количество узлов
    x  : точка, в которой вычисляем значение интерполированного многочлена

    Возвращает:
    значение интерполированного многочлена в точке x
    """
    total = 0
    for i in range(n):
        product = 1
        for j in range(n):
            if i != j:
                product *= (x - xs[j]) / (xs[i] - xs[j])
        total += ys[i] * product
    return total
```

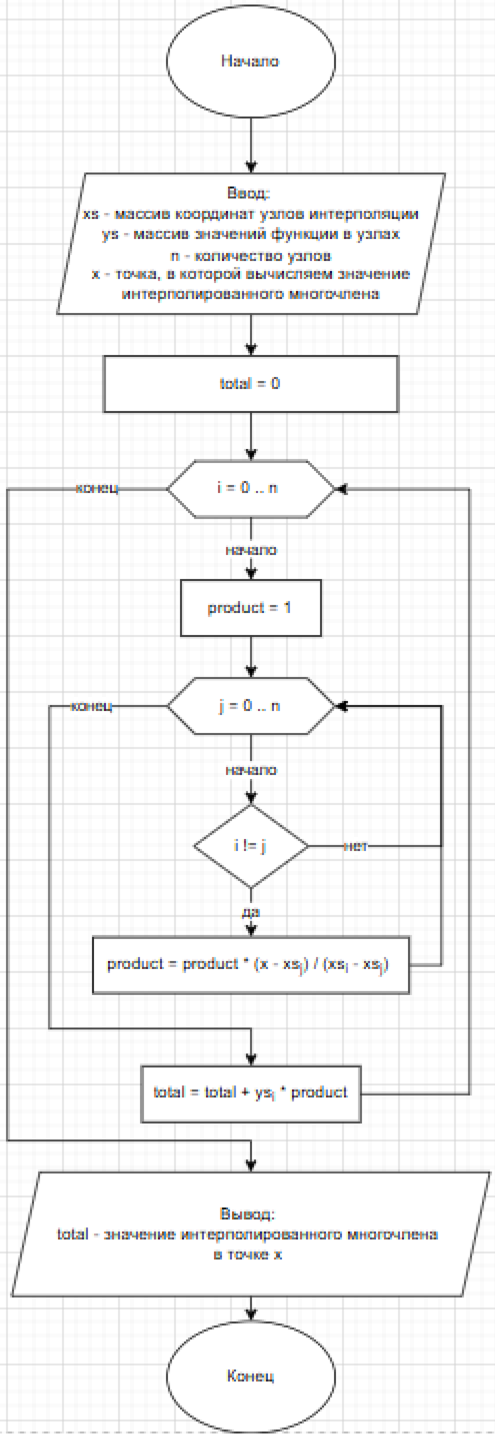
```python
def newton_divided_difference_polynomial(xs, ys, n, x):
    """
    Интерполяция многочленом Ньютона с использованием разделённых разностей.

    Входные параметры:
    xs : массив координат узлов интерполяции
    ys : массив значений функции в узлах
    n  : количество узлов
    x  : точка, в которой вычисляем значение интерполированного многочлена

    Возвращает:
    значение интерполированного многочлена в точке x
    """
    coef = divided_differences(xs, ys)
    total = ys[0]
    for k in range(1, n):
        product = 1
        for j in range(k):
            product *= x - xs[j]
        total += coef[k] * product
    return total

```

```python
def newton_finite_difference_polynomial(xs, ys, n, x):
    """
    Интерполяция многочленом Ньютона с использованием конечных разностей.

    Входные параметры:
    xs : массив координат узлов интерполяции (равномерно расположенных)
    ys : массив значений функции в узлах
    n  : количество узлов
    x  : точка, в которой вычисляем значение интерполированного многочлена

    Возвращает:
    значение интерполированного многочлена в точке x
    """

    h = xs[1] - xs[0]  # шаг
    delta_y = finite_differences(ys)
    total = ys[0]
    for k in range(1, n):
        product = 1
        for j in range(k):
            product *= (x - xs[0]) / h - j
        total += delta_y[0][k] * product / factorial(k)
    return total
```

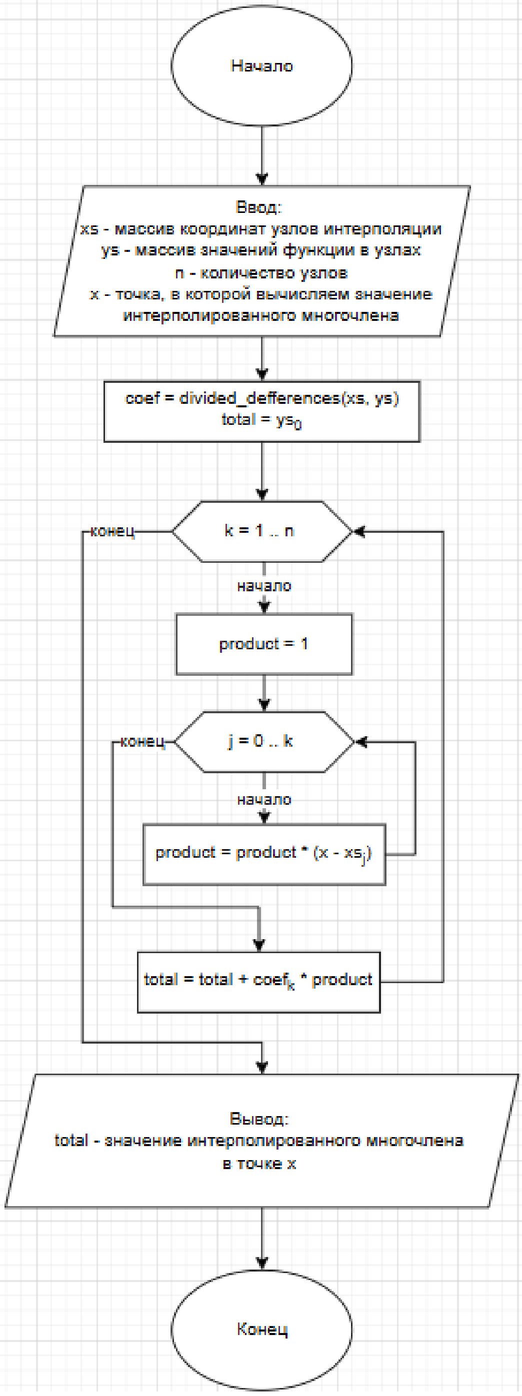
```python
def gauss_polynomial(xs, ys, n, x):
    """
    Интерполяция многочленом Гаусса с использованием центральных конечных разностей.

    Входные параметры:
    xs : массив координат узлов интерполяции (равномерно расположенных)
    ys : массив значений функции в узлах
    n  : количество узлов
    x  : точка, в которой вычисляем значение интерполированного многочлена

    Возвращает:
    значение интерполированного многочлена в точке x
    """

    m = len(xs)
    alpha_ind = (m - 1) // 2  # центральный узел

    # Построение таблицы конечных разностей
    fin_difs = []
    fin_difs.append(ys[:])
    for k in range(1, m):
        prev = fin_difs[-1]
        current = []
        for i in range(len(prev) - 1):
            current.append(prev[i + 1] - prev[i])
        fin_difs.append(current)

    h = xs[1] - xs[0]
    t = (x - xs[alpha_ind]) / h
    total = ys[alpha_ind]

    dts = [0, 1, -1, 2, -2, 3, -3, 4, -4]  # порядок множителей для произведения
    for k in range(1, n):
        product = 1
        for j in range(k):
            product *= t + dts[j]
        mid_index = len(fin_difs[k]) // 2
        if len(fin_difs[k]) % 2 == 0:
            index = mid_index - 1
        else:
            index = mid_index
        total += fin_difs[k][index] * product / factorial(k)

    return total
```

Логические схемы:

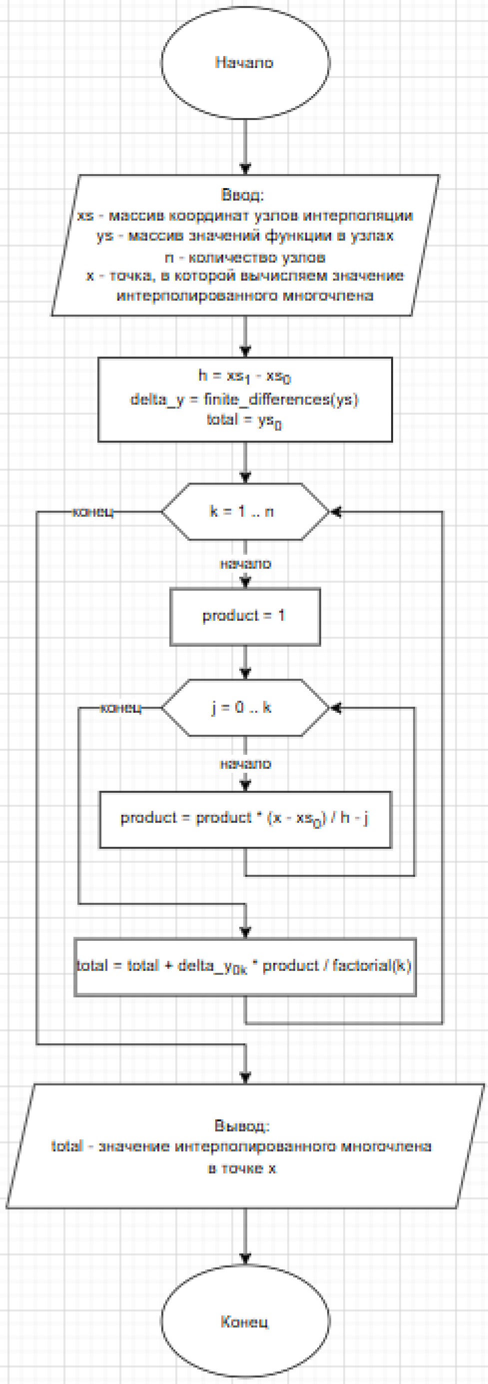
Многочлен Лагранжа



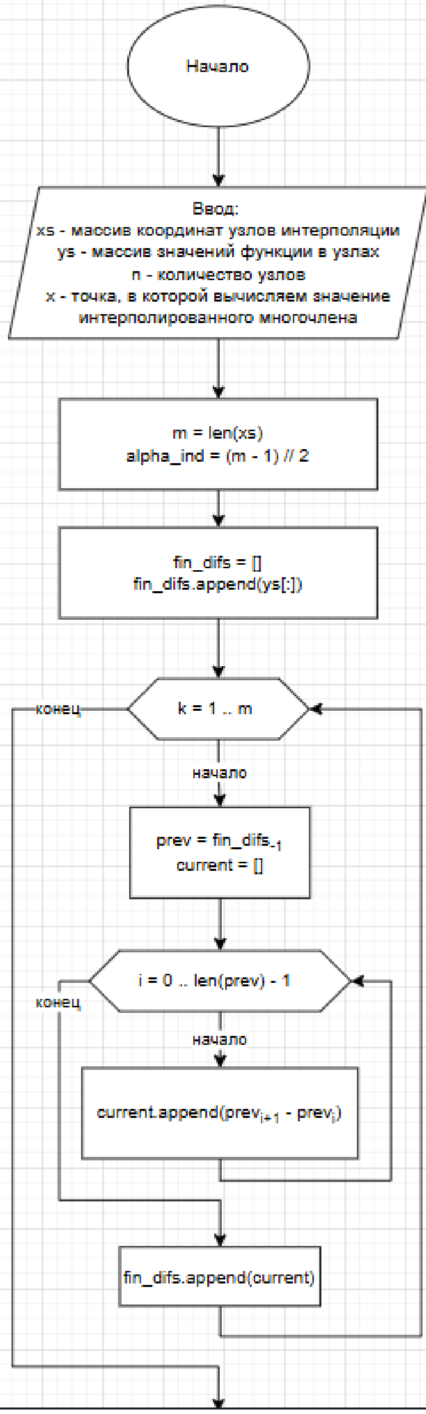
Многочлен Ньютона (разделённые разности)

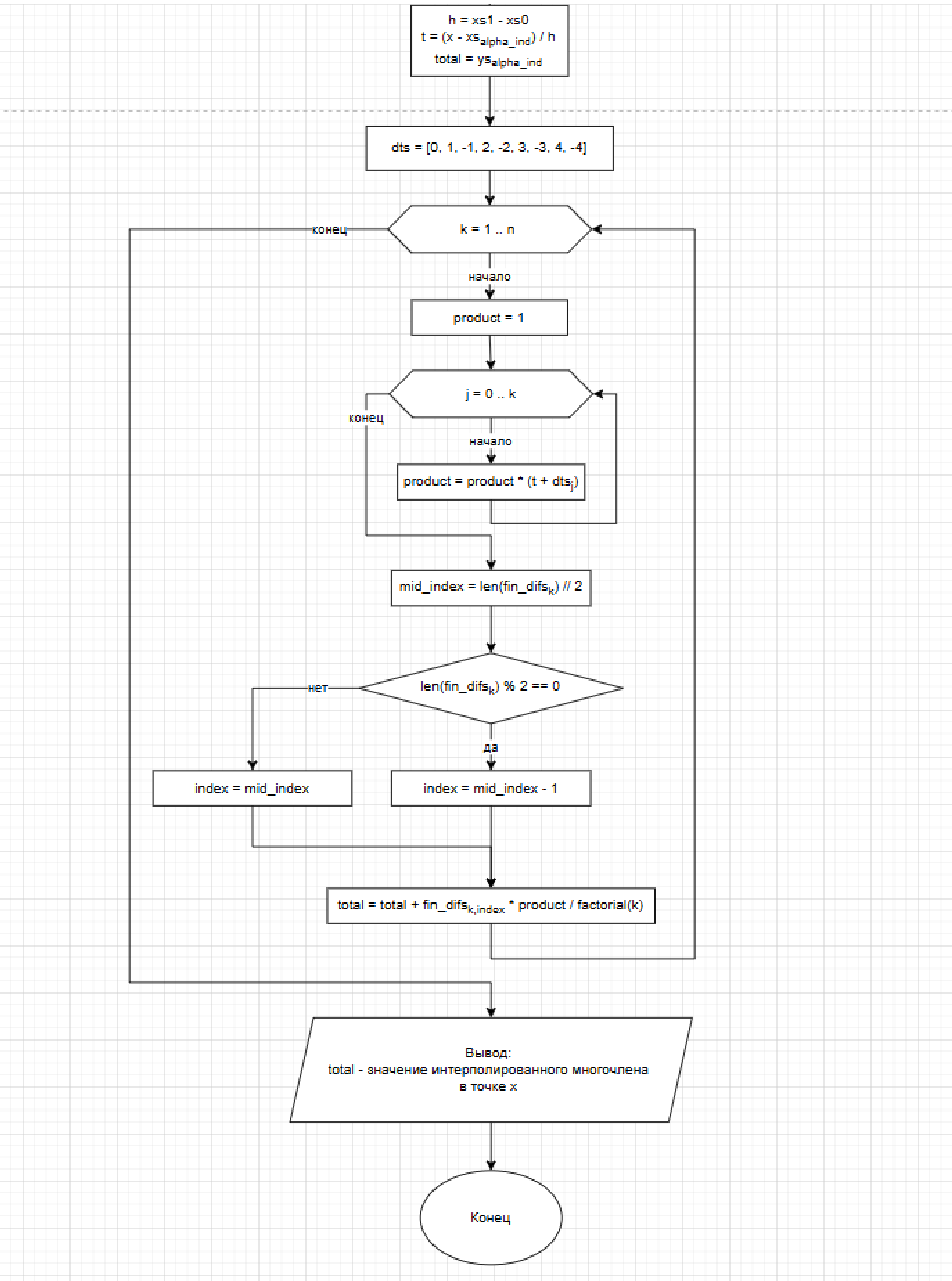


Многочлен Ньютона (конечные разности)

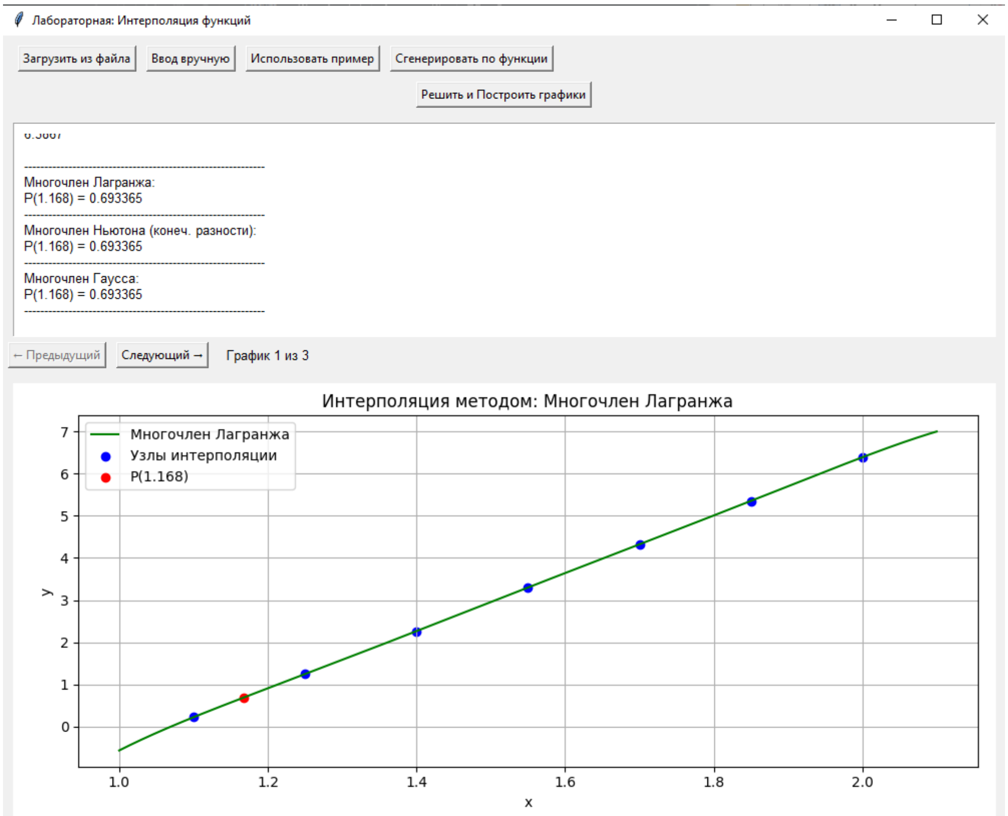


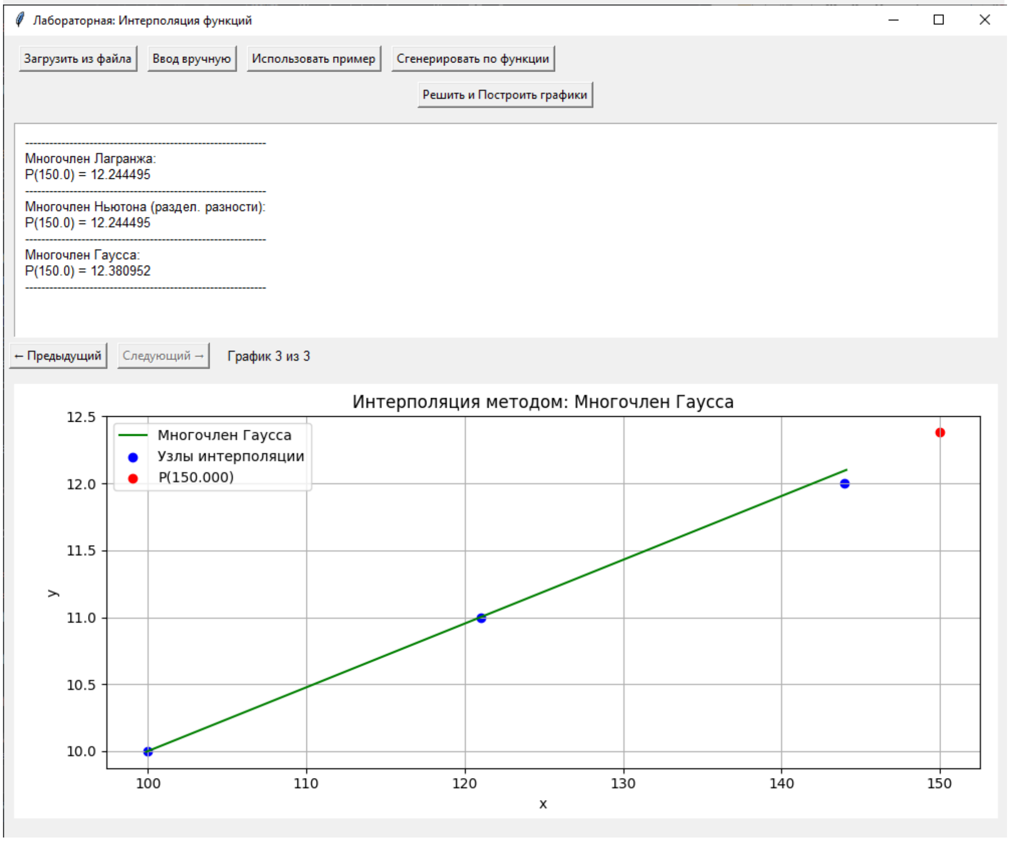
Многочлен Гаусса

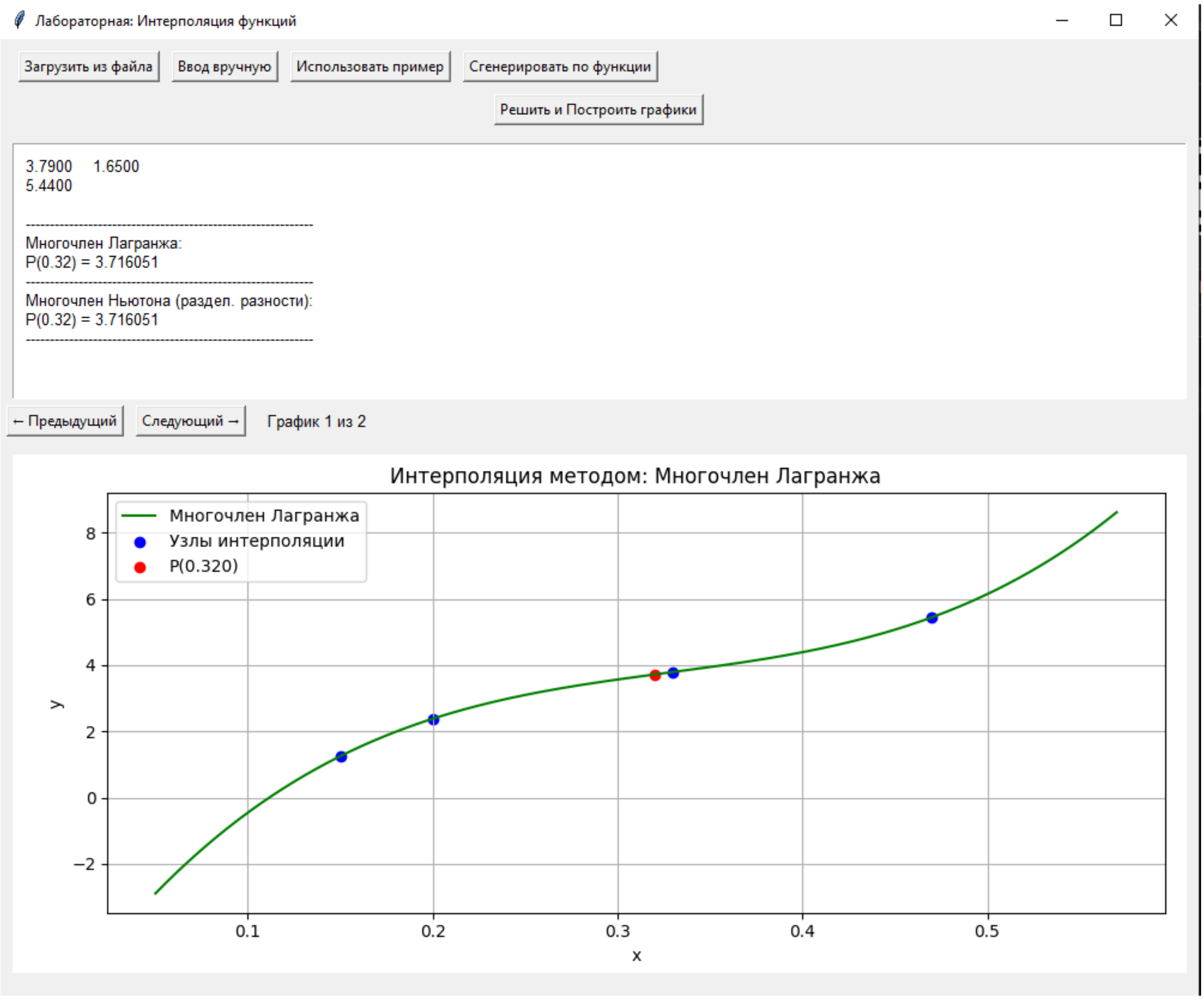




Результаты выполнения программы при различных исходных данных:


Лабораторная: Интерполяция функций
Загрузить из файла
Ввод вручную
Использовать пример
Сгенерировать по функции
Решить и Построить графики
Многочлен Лагранжа:
P(1.168) = 0.693365
Многочлен Ньютона (конеч. разности):
P(1.168) = 0.693365
Многочлен Гаусса:
P(1.168) = 0.693365
← Предыдущий
Следующий →
График 1 из 3
Интерполяция методом: Многочлен Лагранжа
Многочлен Лагранжа
Узлы интерполяции
P(1.168)
y
x


Лабораторная: Интерполяция функций
Загрузить из файла
Ввод вручную
Использовать пример
Сгенерировать по функции
Решить и Построить графики
Многочлен Лагранжа:
P(150.0) = 12.244495
Многочлен Ньютона (раздел. разности):
P(150.0) = 12.244495
Многочлен Гаусса:
P(150.0) = 12.380952
← Предыдущий
Следующий →
График 3 из 3
Интерполяция методом: Многочлен Гаусса
Многочлен Гаусса
Узлы интерполяции
P(150.000)
y
x



Вывод

В ходе выполнения данной лабораторной работы я рассмотрел и реализовал методы интерполяции Ньютона и Гаусса для заданной таблицы данных. Интерполяция позволяет нам предсказывать значения функции в промежуточных точках на основе имеющихся данных.

С помощью разработанной программы были вычислены приближенные значения функции для заданных аргументов с использованием методов Ньютона и Гаусса. Было проведено сравнение результатов, полученных разными методами.

Результаты показали, что оба метода могут быть эффективно использованы для интерполяции, но их точность может зависеть от конкретной функции и распределения данных. Эта работа подчеркивает важность выбора подходящего метода интерполяции в соответствии с требованиями конкретной задачи.