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 № 10

Выполнил:
Мамонтов Г. А.

Преподаватели:
Машина Е. А.
Малышева Т. А.

Санкт-Петербург, 2025 г

Цель лабораторной работы.

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

Порядок выполнения работы.

Вычислительная часть:

1. Выбрать из табл. 1 заданную по варианту таблицу $y = f(x)$ (таблица 1.1 – таблица 1.5);
2. Построить таблицу конечных разностей для заданной таблицы. Таблицу отразить в отчете;
3. Вычислить значения функции для аргумента $X1$ (см. табл.1), используя первую или вторую интерполяционную формулу Ньютона. Обратить внимание какой конкретно формулой необходимо воспользоваться;
4. Вычислить значения функции для аргумента $X2$ (см. табл. 1), используя первую или вторую интерполяционную формулу Гаусса. Обратить внимание какой конкретно формулой необходимо воспользоваться;
5. Подробные вычисления привести в отчете.

Далее написать программу

Рабочие формулы.

Вычислительная часть лабораторной работы.

Выбираем нужную таблицу:

x	y
2.10	3.7587
2.15	4.1861
2.20	4.9218
2.25	5.3487
2.30	5.9275
2.35	6.4193
2.40	7.0839

Строим таблицу конечных разностей

x	y	$\Delta^1 y$	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$	$\Delta^6 y$
2.10	3.7587	0.4274					
2.15	4.1861	0.7357	0.3083				
2.20	4.9218	0.4269	-0.3088	-0.6171			
2.25	5.3487	0.5788	0.1519	0.4607	1.0778		
2.30	5.9275	0.4918	-0.0870	-0.2389	-0.6996	-1.7774	
2.35	6.4193	0.6646	0.1728	0.2598	0.4987	1.1983	2.9757
2.40	7.0839						

Интерполяция по Ньютону

$X_1 = 2.355$

Ближе к **концу таблицы** → используем **формулу Ньютона назад**

Базовая точка: $x_0 = 2.40$; Шаг: $h = 0.05$; $t = \frac{2.355 - 2.40}{0.05} = -0.9$

$$f(x) \approx y_0 + t\Delta_{y-1} + \frac{t(t+1)}{2!}\Delta^2_{y-2} + \dots$$

Подставляем $y_0 = 7.0839$; $\Delta_{y-1} = 0.6646$; $\Delta_{y-2} = -0.0870$

Листинг программы.

```
import math
import matplotlib.pyplot as plt

# Метод Лагранжа: позволяет интерполировать значение функции в точке x,
# используя формулу, в которой каждый член строится на основе всех
других x.
def lagrange_interpolation(x_values, y_values, x):
    result = 0
    n = len(x_values)
    for i in range(n):
        term = y_values[i]  # начинаем с y_i
        for j in range(n):
```

```
            if i != j:
                # умножаем на (x - x_j) / (x_i - x_j) для всех j ≠ i
                # эта часть формирует базисный полином L_i(x)
                term *= (x - x_values[j]) / (x_values[i] - x_values[j])
        result += term  # суммируем все члены: f(x) ≈ Σ y_i * L_i(x)
    return result

# Метод Ньютона (разделённые разности): работает с любыми x, даже если шаг не одинаковый
# Постепенно строится таблица разделённых разностей, используется для построения полинома

def newton_divided_differences(x_values, y_values, x):
    n = len(x_values)
    coef = y_values.copy()  # коэффициенты разделённых разностей, начнем с y_i
    for j in range(1, n):
        for i in range(n - 1, j - 1, -1):
            # формула разделённой разности:
            # f[x_i,...,x_{i-j}] = (f[x_i,...,x_{i-j+1}] - f[x_{i-1},...,x_{i-j}]) / (x_i - x_{i-j})
            coef[i] = (coef[i] - coef[i - 1]) / (x_values[i] - x_values[i - j])
    result = coef[-1]  # начинаем с самого последнего коэффициента
    for i in range(n - 2, -1, -1):
        # полином строится справа налево: P(x) = (...(a_n * (x - x_{n-1}) + a_{n-1})...) + a_0
        result = result * (x - x_values[i]) + coef[i]
    return result

# Метод Ньютона (конечные разности вперёд): требует равномерный шаг между x
# Строим таблицу разностей, используем t = (x - x0) / h

def newton_forward_difference(x_values, y_values, x):
    n = len(x_values)
    h = x_values[1] - x_values[0]  # вычисляем шаг h
    for i in range(1, n - 1):
        if abs((x_values[i + 1] - x_values[i]) - h) > 1e-10:
            raise ValueError("x не равномерны — метод Ньютона с конечными разностями неприменим.")
    diff_table = [y_values.copy()]  # первая строка таблицы — значения функции
```

```
    for i in range(1, n):
        current_diff = []
        for j in range(n - i):
            # вычисляем разности: Δ^k f(x) = Δ^{k-1} f(x_{i+1}) -
Δ^{k-1} f(x_i)
            current_diff.append(diff_table[i - 1][j + 1] - diff_table[i
- 1][j])
        diff_table.append(current_diff)
    t = (x - x_values[0]) / h  # нормализованное расстояние до x0
    result = y_values[0]  # начинаем с f(x0)
    term = 1
    for i in range(1, n):
        # вычисляем t(t - 1)(t - 2)... / i!
        term *= (t - (i - 1)) / i
        # добавляем i-ю разность, умноженную на соответствующий
коэффициент
        result += term * diff_table[i][0]
    return result

# Метод Ньютона (конечные разности назад): аналогично вперёд, но от
последнего узла

def newton_backward_difference(x_values, y_values, x):
    n = len(x_values)
    h = x_values[1] - x_values[0]  # шаг
    for i in range(1, n - 1):
        if abs((x_values[i + 1] - x_values[i]) - h) > 1e-10:
            raise ValueError("x не равномерны — метод назад
неприменим.")
    diff_table = [y_values.copy()]  # начальные значения
    for i in range(1, n):
        current_diff = []
        for j in range(n - i):
            # строим таблицу разностей, как и для метода вперёд
            current_diff.append(diff_table[i - 1][j + 1] - diff_table[i
- 1][j])
        diff_table.append(current_diff)
    t = (x - x_values[-1]) / h  # нормализованное расстояние от
последнего узла
    result = y_values[-1]  # начинаем с f(x_n)
    term = 1
    for i in range(1, n):
        # коэффициенты вида t(t+1)(t+2)... / i!
```

```
        term *= (t + i - 1) / i
        # добавляем соответствующую разность с конца таблицы
        result += term * diff_table[i][-1]
    return result

# --- Ввод данных ---
def input_points_from_keyboard():
    print("Введите количество точек:")
    n = int(input())
    x_values = []
    y_values = []
    print("Введите значения x и y через пробел, по одной паре на
строку:")
    for _ in range(n):
        x, y = map(float, input().split())
        x_values.append(x)
        y_values.append(y)
    return x_values, y_values

def input_points_from_file():
    print("Введите путь к файлу: ")
    filename = input().strip()
    x_values, y_values = [], []
    with open(filename, 'r') as f:
        for line in f:
            x, y = map(float, line.strip().split())
            x_values.append(x)
            y_values.append(y)
    return x_values, y_values

def input_points_from_function():
    print("Выберите функцию:")
    print("1 — sin(x)")
    print("2 — cos(x)")
    print("3 — exp(x)")
    choice = input("Ваш выбор: ")
    func = math.sin
    if choice == '2':
        func = math.cos
    elif choice == '3':
        func = math.exp

    a = float(input("Начало интервала: "))
```

```
    b = float(input("Конец интервала: "))
    n = int(input("Количество точек: "))
    x_values = [a + i * (b - a) / (n - 1) for i in range(n)]
    y_values = [func(x) for x in x_values]
    return x_values, y_values

# --- Таблица конечных разностей ---
def print_forward_difference_table(x_values, y_values):
    n = len(x_values)
    table = [y_values.copy()]
    for i in range(1, n):
        row = [table[i - 1][j + 1] - table[i - 1][j] for j in range(n -
i)]
        table.append(row)
    print("\nТаблица конечных разностей:")
    for i in range(n):
        print(f"{x_values[i]:>6.3f} |", end='')
        for j in range(n - i):
            print(f" {table[j][i]:>10.6f}", end='')
        print()

# --- Графики ---
def plot_interpolation(x_values, y_values, method_func, label):
    xs = [x_values[0] + i * 0.01 for i in range(int((x_values[-1] -
x_values[0]) / 0.01) + 1)]
    ys = [method_func(x_values, y_values, x) for x in xs]
    plt.plot(xs, ys, label=label)

# --- Главная логика ---
def main():
    print("Выберите способ ввода данных:")
    print("1 — ввод с клавиатуры")
    print("2 — ввод из файла")
    print("3 — по заданной функции")
    choice = input("Ваш выбор: ")

    if choice == '1':
        x_values, y_values = input_points_from_keyboard()
    elif choice == '2':
        x_values, y_values = input_points_from_file()
    elif choice == '3':
        x_values, y_values = input_points_from_function()
    else:
```

```
        print("Неверный выбор.")
        return

    print_forward_difference_table(x_values, y_values)

    x = float(input("Введите значение x, в котором нужно
интерполировать: "))

    try:
        l = lagrange_interpolation(x_values, y_values, x)
        print(f"Значение по Лагранжу: {l:.6f}")
    except Exception as e:
        print(f"Ошибка Лагранжа: {e}")

    try:
        nd = newton_divided_differences(x_values, y_values, x)
        print(f"Ньютон (разделённые): {nd:.6f}")
    except Exception as e:
        print(f"Ошибка Ньютона (разделённые): {e}")

    try:
        nf = newton_forward_difference(x_values, y_values, x)
        print(f"Ньютон (вперёд): {nf:.6f}")
    except Exception as e:
        print(f"Ошибка Ньютона (вперёд): {e}")

    try:
        nb = newton_backward_difference(x_values, y_values, x)
        print(f"Ньютон (назад): {nb:.6f}")
    except Exception as e:
        print(f"Ошибка Ньютона (назад): {e}")

    # Построение графиков
    plt.figure(figsize=(10, 6))
    plt.scatter(x_values, y_values, color='black', label='Узлы')
    try:
        plot_interpolation(x_values, y_values, lagrange_interpolation,
'Лагранж')
    except:
        pass
    try:
        plot_interpolation(x_values, y_values,
newton_divided_differences, 'Ньютон (разделённые)')
```

```
    except:
        pass
    try:
        plot_interpolation(x_values, y_values,
newton_forward_difference, 'Ньютон (вперёд)')
    except:
        pass
    try:
        plot_interpolation(x_values, y_values,
newton_backward_difference, 'Ньютон (назад)')
    except:
        pass

    plt.title("Интерполяционные многочлены")
    plt.xlabel("x")
    plt.ylabel("f(x)")
    plt.grid(True)
    plt.legend()
    plt.show()

# Запуск программы
if __name__ == '__main__':
    main()
```

Untitled8.ipynb

Результаты выполнения программы.

Ввод:
1.0 0.8415
1.1 0.8912
1.2 0.932
1.3 0.9636
1.4 0.9854

```
Выберите способ ввода данных:
1 — ввод с клавиатуры
2 — ввод из файла
3 — по заданной функции
Ваш выбор: 2
Введите путь к файлу:
/content/sample_data/teat1

Таблица конечных разностей:
 1.000 |   0.841500   0.049700  -0.008900  -0.000300  -0.000300
 1.100 |   0.891200   0.040800  -0.009200  -0.000600
 1.200 |   0.932000   0.031600  -0.009800
 1.300 |   0.963600   0.021800
 1.400 |   0.985400
Введите значение x, в котором нужно интерполировать: 1.35
Значение по Лагранжу: 0.975774
Ньютон (разделённые): 0.975774
Ньютон (вперёд): 0.975774
Ньютон (назад): 0.975774
```

Ввод:
2.0 7.3891
2.1 8.1662
2.2 9.025
2.3 9.9742
2.4 11.023

```
Выберите способ ввода данных:
1 — ввод с клавиатуры
2 — ввод из файла
3 — по заданной функции
Ваш выбор: 2
Введите путь к файлу:
/content/sample_data/test2

Таблица конечных разностей:
 2.000 |   7.389100   0.777100   0.081700   0.008700   0.000500
 2.100 |   8.166200   0.858800   0.090400   0.009200
 2.200 |   9.025000   0.949200   0.099600
 2.300 |   9.974200   1.048800
 2.400 |  11.023000
Введите значение x, в котором нужно интерполировать: 2.333
Значение по Лагранжу: 10.308828
Ньютон (разделённые): 10.308828
Ньютон (вперёд): 10.308828
Ньютон (назад): 10.308828
```

Ввод:
0.0 1.0
0.5 0.8776
1.0 0.5403
1.5 0.0707
2.0 -0.4161

```
Выберите способ ввода данных:
1 — ввод с клавиатуры
2 — ввод из файла
3 — по заданной функции
Ваш выбор: 2
Введите путь к файлу:
/content/sample_data/test3

Таблица конечных разностей:
 0.000 |   1.000000  -0.122400  -0.214900   0.082600   0.032500
 0.500 |   0.877600  -0.337300  -0.132300   0.115100
 1.000 |   0.540300  -0.469600  -0.017200
 1.500 |   0.070700  -0.486800
 2.000 |  -0.416100
Введите значение x, в котором нужно интерполировать: 0.777
Значение по Лагранжу: 0.712751
Ньютон (разделённые): 0.712751
Ньютон (вперёд): 0.712751
Ньютон (назад): 0.712751
```

Выводы