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

Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа №5

Вариант 3

Группа: P3269

Выполнили:

Грибкова В.Е

Долганова О.А

Проверил:

Машина Е. А.

Г. Санкт-Петербург

1. Цель лабораторной работы: решить задачу интерполяции, найти значения функции при заданных значениях аргумента, отличных от узловых точек.
2. Рабочие формулы используемых методов.

$\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i$ - конечные разности k-го порядка

Интерполяционные формулы Ньютона для равноотстоящих узлов

Введем обозначение: $t = (x - x_0)/h$. Тогда получим формулу Ньютона, которая называется **первой интерполяционной формулой Ньютона для интерполирования вперед:**

$$N_n(x) = y_0 + t\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + \cdots + \frac{t(t-1)\ldots(t-n+1)}{n!}\Delta^n y_0$$

Полученное выражение может аппроксимировать функцию на всем отрезке изменения аргумента $[x_0, x_n]$. Однако более целесообразно (с точки зрения повышения точности расчетов) использовать эту формулу для $x_0 \le x \le x_1$. При этом за x_0 может приниматься любой узел интерполяции x_k. Например, для $x_1 \le x \le x_2$, вместо x_0 надо взять значение x_1. Тогда интерполяционный многочлен Ньютона:

$$N_n(x) = y_i + t\Delta y_i + \frac{t(t-1)}{2!}\Delta^2 y_i + \cdots + \frac{t(t-1)\ldots(t-n+1)}{n!}\Delta^n y_i \quad (*)$$

Интерполяционную формулу (*) обычно используют для вычислений значений функции в точках левой половины отрезка.

3. Вычислительная часть

	x	y	№ варианта	X_1	X_2
Таблица 1.3	1,10	0,2234	3	1,121	1,482
	1,25	1,2438	8	1,852	1,652
	1,40	2,2644	13	1,168	1,463
	1,55	3,2984	18	1,875	1,575
	1,70	4,3222	23	1,189	1,491
	1,85	5,3516	28	1,891	1,671
	2,00	6,3867	33	1,217	1,473

2. Построим таблицу конечных разностей для заданной таблицы.

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	1,0210	1,0340	1,0238	1,0502	1,1253
1,25	1,2438	2,0414	2,0550	2,0578	2,0738	2,2005	

1,40	2,2644	4,0964	4,1128	4,1316	4,2743		
1,55	3,2984	8,1952	8,2244	8,4060			
1,70	4,3222	16,4196	16,6304				
1,85	5,3516	32.9820					
2,00	6,3867						

3. Вычислим значения функции для аргумента X1=1,121, используя первую интерполяционную формулу Ньютона.

$t = \frac{(x-x_0)}{h} = \frac{1{,}121-1{,}1}{0{,}15} = 0,14$

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

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

$y(1,121) \approx 0,3715$

4. Вычислим значения функции для аргумента X2=1,482, используя вторую интерполяционную формулу Гаусса.

x_i	y_i	$\Delta\ y_i$	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^4 y_i$	$\Delta^5 y_i$	$\Delta^6 y_i$
1,10	0,2234	1,0204	1,0210	1,0340	1,0238	1,0502	1,1253
1,25	1,2438	2,0414	2,0550	2,0578	2,0738	2,2005	
1,40	2,2644	4,0964	4,1128	4,1316	4,2743		
x0=1,55	3,2984	8,1952	8,2244	8,4060			
1,70	4,3222	16,4196	16,6304				
1,85	5,3516	32.9820					
2,00	6,3867						

$t = \frac{(x-x_0)}{h} = \frac{1{,}428-1{,}55}{0{,}15} = -0,813$

$P_6(x) = y0\ + t\Delta y_{-1} + \frac{t(t+1)}{2!}\Delta^2 y_{-1} + \frac{t(t+1)(t-2)}{3!}\Delta^3 y_{-2} + \frac{t(t+1)(t-3)}{4!}\Delta^4 y_{-2} + \frac{t(t+1)(t-3)(t-4)}{5!}\Delta^5 y_{-3} +$

$+ \frac{t(t+1)(t-3)(t-4)(t-5)}{6!}\Delta^6 y_{-3}$

$y(1,428) \approx 2{,}830$

4. Листинг программы.

```
import math
import numpy as np
from matplotlib import pyplot as plt

# Функция для выбора режима ввода данных
def choose_input_mode():
    while True:
        try:
            choice = int(input("Введите 1, если ввод данных будет
происходить из файла. Введите 2, если с клавиатуры. "
                               "Введите 3 для выбора уравнения"))
            if choice not in [1, 2, 3]:
                print('Пожалуйста, введите 1, 2 или 3')
                continue
            else:
                return choice
        except ValueError:
            print("Пожалуйста, введите число")
            continue

# Функция для получения данных
def get_data():
    mode = choose_input_mode()
    while True:
        try:
            x_values = []
            y_values = []
            if mode == 1:
                # Чтение данных из файла
                with open('input2.txt', 'r') as file:
                    while (line := file.readline()) != '\n':
                        x_values.append(float(line.split(' ')[0]))
                        y_values.append(float(line.split(' ')[1]))
                    return [x_values, y_values]
            elif mode == 2:
                # Ввод данных с клавиатуры
                print('Введите координаты')
                while (line := input()) != '':
                    x_values.append(float(line.split(' ')[0]))
                    y_values.append(float(line.split(' ')[1]))
                return [x_values, y_values]
            else:
                # Выбор уравнения
                print('1. sin(x)')
                print('2. x ** 2')
                print('Выберите уравнение (1 или 2)')
                equation_choice = int(input())
```

```
                print('Введите исследуемый интервал')
                interval = input().split(' ')
                a, b = int(interval[0]), int(interval[1])
                print('Введите количество точек на интервале')
                num_points = int(input())
                for i in range(num_points):
                    x_i = a + (b - a) * i / num_points
                    x_values.append(x_i)
                    if equation_choice == 1:
                        y_values.append(math.sin(x_i))
                    elif equation_choice == 2:
                        y_values.append(x_i ** 2)
                    else:
                        print("Вы ввели неверную цифру")
                return [x_values, y_values]
        except (TypeError, ValueError, IndexError):
            print("Вы ввели неверные данные")
            exit(0)

# Функция для вычисления полинома Лагранжа
def lagrange_polynomial(x_values, y_values, x_current):
    result = 0
    for i in range(len(x_values)):
        term = 1
        for j in range(len(y_values)):
            if i != j:
                term *= (x_current - x_values[j]) / (x_values[i] -
x_values[j])
        result += term * y_values[i]
    return result

# Функция для вычисления коэффициентов Ньютона
def newton_coefficient(x_values, y_values):
    m = len(x_values)
    x = np.copy(x_values)
    y = np.copy(y_values)
    for k in range(1, m):
        y[k:m] = (y[k:m] - y[k - 1]) / (x[k:m] - x[k - 1])
    return y

# Функция для вычисления полинома Ньютона
def newton_polynomial(x_values, y_values, x_current):
    coefficients = newton_coefficient(x_values, y_values)
    n = len(x_values) - 1
    result = coefficients[n]
    for k in range(1, n + 1):
        result = coefficients[n - k] + (x_current - x_values[n -
k]) * result
```

```
    return result

# Вспомогательная функция для вычисления значений t
def t_calculation(t, n, forward=True):
    result = t
    for i in range(1, n):
        if forward:
            result *= t - i
        else:
            result *= t + i
    return result

# Функция для интерполяции с помощью конечных разностей Ньютона
def newton_interpolation(x_values, y_values, x_current):
    n = len(x_values)
    is_equally_spaced = True
    h = x_values[1] - x_values[0]
    for i in range(1, n - 1):
        if round(x_values[i + 1] - x_values[i], 3) != h:
            is_equally_spaced = False
            break
    if not is_equally_spaced:
        return 'Узлы не являются равноотстоящими'
    differences = [[0] * n for _ in range(n)]
    for i in range(n):
        differences[i][0] = y_values[i]
    for i in range(1, n):
        for j in range(n - i):
            differences[j][i] = differences[j + 1][i - 1] -
differences[j][i - 1]
    if x_current <= x_values[n // 2]:
        x0 = n - 1
        for i in range(n):
            if x_current <= x_values[i]:
                x0 = i - 1
                break
        if x0 < 0:
            x0 = 0
        t = (x_current - x_values[x0]) / h
        result = differences[x0][0]
        for i in range(1, n):
            result += (t_calculation(t, i) * differences[x0][i]) /
math.factorial(i)
    else:
        t = (x_current - x_values[n - 1]) / h
        result = differences[n - 1][0]
        for i in range(1, n):
```

```
            result += (t_calculation(t, i, False) * differences[n -
i - 1][i]) / math.factorial(i)
    return result

# Функция для интерполяции с помощью многочлена Стирлинга
def stirling_interpolation(x_values, y_values, x_current):
    is_equally_spaced = True
    h = round(x_values[1] - x_values[0], 3)
    n = len(x_values)
    for i in range(1, n - 1):
        if round(x_values[i + 1] - x_values[i], 3) != h:
            is_equally_spaced = False
            break
    if not is_equally_spaced:
        return 'Узлы не являются равноотстоящими'
    differences = [[0] * n for _ in range(n)]
    for i in range(n):
        differences[i][0] = y_values[i]
    for i in range(1, n):
        for j in range(n - i):
            differences[j][i] = differences[j + 1][i - 1] -
differences[j][i - 1]
    if n % 2 == 0:
        x0 = int(n / 2 - 1)
    else:
        x0 = int(n / 2)
    t = (x_current - x_values[x0]) / h
    if abs(t) > 0.25:
        print('t > 0.25 => результат Стирлинга может быть
неточным')
    result = differences[x0][0]
    comp_t1 = t
    comp_t2 = t**2
    prev_number = 0
    for i in range(1, len(x_values)):
        if i % 2 == 0:
            result += (comp_t2 / math.factorial(i)) *
differences[x0 - (i // 2)][i]
            comp_t2 *= (t**2 - prev_number**2)
        else:
            result += (comp_t1 / math.factorial(i)) *
((differences[x0 - ((i + 1) // 2)][i] + differences[x0 - (((i + 1)
// 2) - 1)][i]) / 2)
            prev_number += 1
            comp_t1 *= (t**2 - prev_number**2)
    return result

# Функция для интерполяции с помощью многочлена Бесселя
```

```
def bessel_interpolation(x_values, y_values, x_current):
    is_equally_spaced = True
    h = round(x_values[1] - x_values[0], 3)
    n = len(x_values)
    differences = [[0] * n for _ in range(n)]
    for i in range(n):
        differences[i][0] = y_values[i]
    for i in range(1, n):
        for j in range(n - i):
            differences[j][i] = differences[j + 1][i - 1] -
differences[j][i - 1]
    for i in range(1, n - 1):
        if round(x_values[i + 1] - x_values[i], 3) != h:
            is_equally_spaced = False
            break
    if not is_equally_spaced:
        return 'Узлы не являются равноотстоящими'
    if n % 2 == 0:
        x0 = int(n / 2 - 1)
    else:
        x0 = int(n / 2)
    t = (x_current - x_values[x0]) / h
    if abs(t) < 0.25 or abs(t) > 0.75:
        print('t < 0.25 или t > 0.75 => результат Бесселя может
быть неточным')
    result = (differences[x0][0] + differences[x0 + 1][0]) / 2
    result += (t - 0.5) * differences[x0][1]
    comp_t = t
    last_number = 0
    for i in range(2, len(x_values)):
        if i % 2 == 0:
            last_number += 1
            comp_t *= (t - last_number)
            result += (comp_t / math.factorial(i)) *
((differences[x0 - i // 2][i] + differences[x0 - ((i // 2) -
1)][i]) / 2)
        else:
            result += (comp_t * (t - 0.5) / math.factorial(i)) *
differences[x0 - ((i - 1) // 2)][i]
            comp_t *= (t + last_number)
    return result

# Функция для вычисления конечных разностей
def finite_differences(data, y_values):
    temp = []
    if len(y_values) <= 1:
        return data
    for i in range(len(y_values) - 1):
```

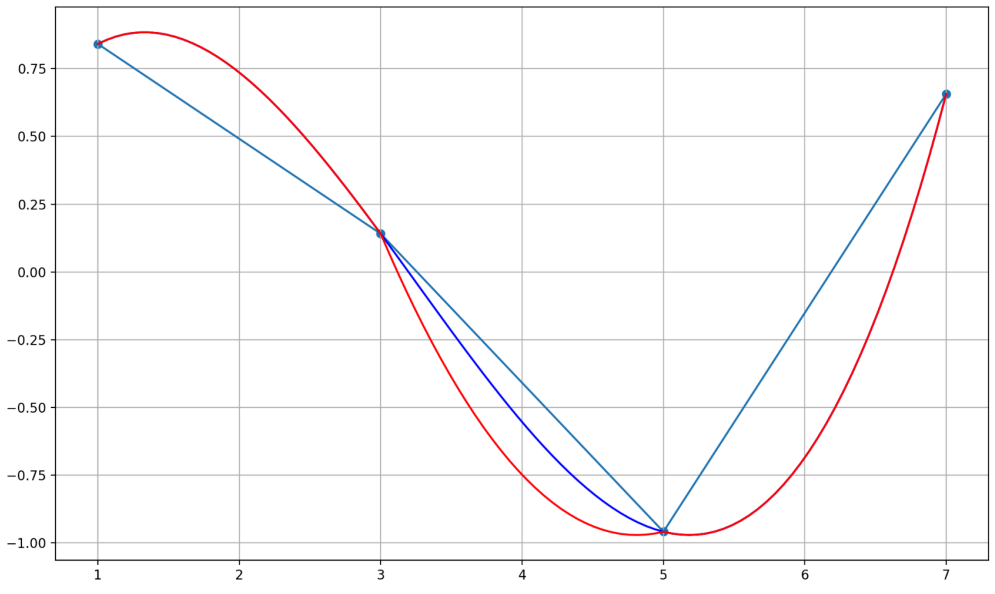
```
        temp.append(y_values[i + 1] - y_values[i])
    data.append(temp)
    return finite_differences(data, temp)

# Главная функция
def main():
    data = get_data()
    print('Конечные разности:', finite_differences([], data[1]))
    x_values = data[0]
    y_values = data[1]
    if len(x_values) != len(set(x_values)):
        temp = []
        for i in range(len(x_values)):
            if x_values[i] not in temp:
                temp.append(x_values[i])
            else:
                temp.append(x_values[i]+0.01)
        x_values = temp
    print('Введите значение аргумента')
    x_current = float(input())
    answer1 = lagrange_polynomial(x_values, y_values, x_current)
    answer2 = newton_polynomial(x_values, y_values, x_current)
    answer3 = newton_interpolation(x_values, y_values, x_current)
    answer4 = stirling_interpolation(x_values, y_values, x_current)
    answer5 = bessel_interpolation(x_values, y_values, x_current)
    print('Полином Лагранжа дал ответ: ', answer1)
    print('Полином Ньютона с разделенными разностями дал ответ: ',
answer2)
    print('Полином Ньютона с конечными разностями дал ответ: ',
answer3)
    print('Многочлен Стирлинга дал ответ: ', answer4)
    print('Многочлен Бесселя дал ответ: ', answer5)
    plt.plot(x_values, y_values)
    plt.scatter(x_values, y_values, label="Вводные точки")
    plt.grid(True)
    plot_x = np.linspace(np.min(x_values), np.max(x_values), 100)
    plot_y = [newton_polynomial(x_values, y_values, x_c) for x_c in
plot_x]
    plt.plot(plot_x, plot_y, color='b', label='Newton_first')
    if answer3 != 'Узлы не являются равноотстоящими':
        plot_y = [newton_interpolation(x_values, y_values, x_c) for
x_c in plot_x]
        plt.plot(plot_x, plot_y, color='r', label='Newton_second')
    plt.show()

# Запуск главной функции
main()
```

5. Результаты выполнения программы

```
Введите 1, если ввод данных будет происходить из файла. Введите 2,
если с клавиатуры. Введите 3 для выбора уравнения3
1. sin(x)
2. x ** 2
Выберите уравнение (1 или 2)
1
Введите исследуемый интервал
1 9
Введите количество точек на интервале
4
Конечные разности: [[-0.7003509767480293, -1.1000442827230057,
1.6159108733819276], [-0.39969330597497643, 2.7159551561049335],
[3.115648462079911]]
Введите значение аргумента
3
t < 0.25 или t > 0.75 => результат Бесселя может быть неточным
Полином Лагранжа дал ответ:  0.1411200080598672
Полином Ньютона с разделенными разностями дал ответ:
0.14112000805986724
Полином Ньютона с конечными разностями дал ответ:
0.14112000805986724
Многочлен Стирлинга дал ответ:  0.1411200080598672
Многочлен Бесселя дал ответ:  0.14112000805986724
```



6. Выводы