Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Курский государственный университет»

Кафедра программного обеспечения и администрирования информационных систем

Направление подготовки математическое обеспечение и администрирование информационных систем

Форма обучения очная

**Отчет**

**по лабораторной работе №6**

«Интерполяционный многочлен Ньютона»

дисциплина «Методы вычислений»

Выполнил:

студент группы 313.1 Козявин М.С.

Проверил:

доцент кафедры АГиТОМ Селиванова И. В.

Курск, 2023

**Цель:** Изучение интерполяционного многочлена Ньютона для равноотстоящих узлов.

**Задание:**

1. Разработать алгоритм и программу вычисления значения функции в промежуточной точке по экспериментальным данным для равноотстоящих узлов.

**Код программы:**

import math

import matplotlib.pyplot as plt

data = [

    [-9, 5],

    [-4, 2],

    [-1, -2],

    [7, 9]

]

*# data = [*

*#     [1.257, 3.081],*

*#     [1.524, 3.837],*

*#     [1.728, 3.648],*

*#     [1.849, 3.424],*

*#     [1.867, 3.175],*

*#     [1.768, 2.910],*

*#     [1.547, 2.638],*

*#     [1.215, 2.369],*

*#     [0.798, 2.109],*

*#     [0.339, 1.864],*

*#     [-0.104, 1.637]*

*# ]*

coef = [[0 for \_\_ in range(len(data))] for \_ in range(len(data))]

*def* calc\_f(*j*, *k*):

    d = data[k][0] - data[j][0]

    if j == k:

        return data[k][1]

    else:

        return (calc\_f(j+1, k)-calc\_f(j, k-1))/d

*def* calc\_x(*n*):

    r = [-data[0][0], 1]

    for x in range(1, n+1):

        a = [-data[x][0], 1]

        l = x+2

        tmp = [0 for \_ in range(l)]

        for i in range(len(r)):

            for j in range(2):

                tmp[i + j] += r[i] \* a[j]

        r = tmp

    return r

*def* calc\_newton(*n*):

    c = [0 for \_ in range(len(data))]

    fc = calc\_f(0, 0)

    c[0] += fc

    for i in range(1, len(data)):

        fc = calc\_f(0, i)

        xc = calc\_x(i-1)

        print("fc:", fc, "xc", xc)

        for j in range(len(xc)):

            c[j] += xc[j]\*fc

    return c

*def* f(*x*, *nums*):

    res = 0

    for i in range(len(nums)):

        res += nums[i]\*math.pow(x, i)

    return res

*def* draw\_graph(*nums*, *size*=100, *dots*=False, *style*="solid"):

    xs = [x / size for x in range(-size\*10, size\*10)]

    ys = [f(x, nums) for x in xs]

    plt.plot(xs, ys, *linestyle*=style)

    if dots:

        for i in range(len(data)):

            plt.scatter(data[i][0], data[i][1], *color*="black")

r = calc\_newton(len(data))

print(f(1.02, r))

draw\_graph(r, 100, True)

plt.grid()

plt.show()

**Тестирование:**



