Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра Систем Штучного Інтелекту

Звіт

до лабораторної роботи № 6

з дисципліни

Чисельні методи

на тему:

“Побудова інтерполяційного полінома Лагранжа та Ньютона”

Варіант №24(Поліном Ньютона)

Виконав: студент КН-217

Ратушняк Денис

Прийняла: доцент каф. СШІ

Мочурад Л. І.

Львів – 2022

Мета роботи: – набути навиків вирішення задач інтерполювання на основі побудови інтерполяційних поліномів Лагранжа та Ньютона

Завдання.

Побудувати інтерполяційний поліном Лагранжа(для нерівновіддалених вузлів) або **Ньютона(для рівновіддалених вузлів)** для функції, що задана таблично. Побудувати графік заданих точок та отриманого поліному.

8

x	3	4	5	6	7
y	12	-1	5	8	3

Програмна реалізація на мові програмування python:

```
import math
from math import pi, sin, cos
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
from numpy import linspace

plt.style.use('seaborn-v0_8')
matplotlib.use('TkAgg')


def divided_diff(x, y):
    '''
    function to calculate the divided
    differences table
    '''
    n = len(y)
    coef = np.zeros([n, n])
    # the first column is y
    coef[:, 0] = y

    for j in range(1, n):
        for i in range(n - j):
            coef[i][j] = \
                (coef[i + 1][j - 1] - coef[i][j - 1]) / (x[i + j] - x[i])

    return coef


def newton_poly(coef, x_data, x):
    '''
    evaluate the newton polynomial
    at x
    '''
    n = len(x_data) - 1
    p = coef[n]
    for k in range(1, n + 1):
        p = coef[n - k] + (x - x_data[n - k]) * p
    return p


def calc_euclid_norm(ar1, ar2):
    error = 0
```

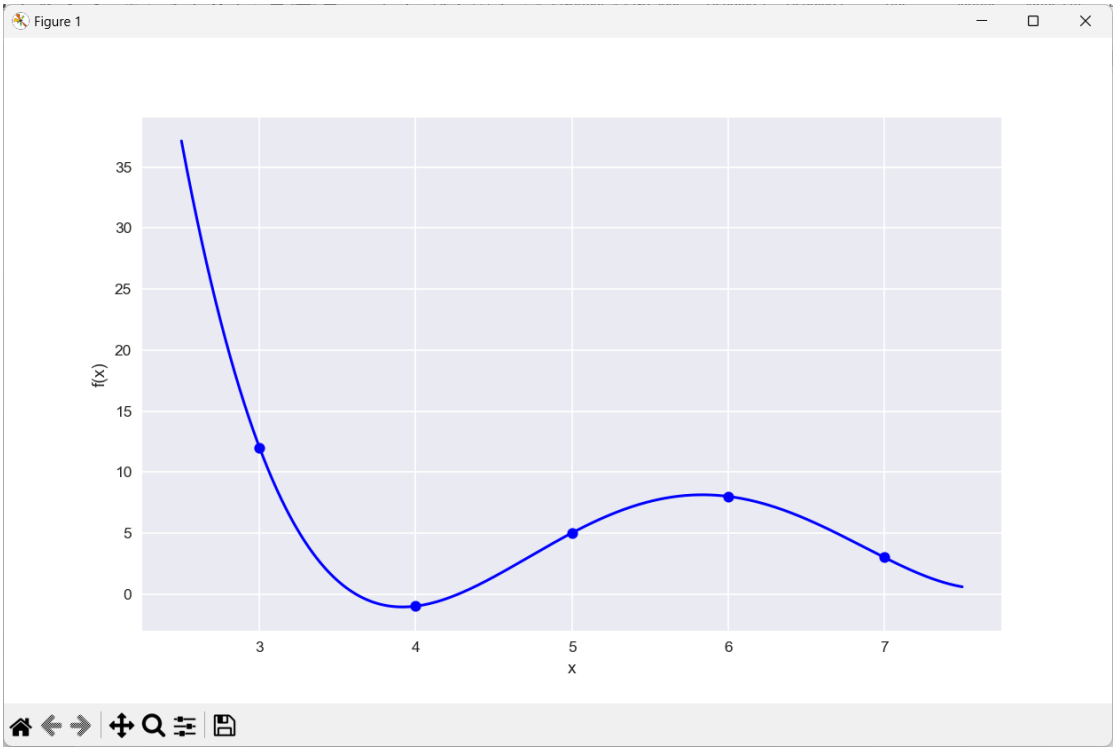
```
    for i in range(len(ar1)):
        error += (ar1[i] - ar2[i]) ** 2
    return math.sqrt(error)


def print_polynom_for_given_cnt_of_points(ddc):
    print("Newton`s polynom: ")
    print(ddc[0], end=' ')
    for i in range(1, len(ddc)):
        if ddc[i] >= 0:
            print('+', end=' ')
        else:
            print('-', end=' ')
        print(abs(ddc[i]), end='')
        for j in range(i):
            print('(x-x', j, ')', sep='', end='')
        print(end=' ')
    print()

    print("Newton`s polynom with given data: ")
    print(ddc[0], end=' ')
    for i in range(1, len(ddc)):
        if ddc[i] >= 0:
            print('+', end=' ')
        else:
            print('-', end=' ')
        print(abs(ddc[i]), end='')
        for j in range(i):
            if x_test[j] < 0:
                print('(x+', abs(x_test[j]), ')', sep='', end='')
            else:
                print('(x-', x_test[j], ')', sep='', end='')
        print(end=' ')


def do_things_for_given_functions():

    print_polynom_for_given_cnt_of_points(ddc)

    y_for_error = newton_poly(ddc, x_test, x)

    error = calc_euclid_norm(y, y_for_error)

    print("\nError:", math.sqrt(error))


print("Please input 1 for default task, 2 for y = -3x^3 - 5x^2 + 3x + 5, 3 for
y = sin(x) + cos(x)^3")

type = int(input())

if type == 1:
    x_test = np.array([3, 4, 5, 6, 7])
    y_test = np.array([12, -1, 5, 8, 3])

    # get the divided difference coef
    ddc = divided_diff(x_test, y_test)[0, :]

    # evaluate on new data points
    x1 = linspace(2.5, 7.5, 1000)
    y_new = newton_poly(ddc, x_test, x1)
    y = y_test
    x = x_test
```

```
    cnt_points = 4
    do_things_for_given_functions()

    plt.figure(figsize=(10, 6))
    plt.xlabel('x')
    plt.ylabel('f(x)')

    plt.plot(x1, y_new, 'b')
    plt.plot(x_test, y_test, 'bo')

    plt.show()


elif type == 2:
    def fx1(x):
        return -3 * (x ** 3) - 5 * (x ** 2) + 3 * x + 5


    x_test0 = []
    print("Input number of test points:")
    cnt_points = int(input())
    cnt_points -= 1
    left = -2
    right = 2

    step = (right - left) / cnt_points

    while True:
        x_test0.append(left)
        left += step
        if left > right:
            break

    x_test = np.array(x_test0)
    y_test = np.array([fx1(x) for x in x_test])

    ddc = divided_diff(x_test, y_test)[0, :]

    x = linspace(-2, 2, 1000)
    y = -3 * (x ** 3) - 5 * (x ** 2) + 3 * x + 5
    y_new = newton_poly(ddc, x_test, x)

    do_things_for_given_functions()

    plt.xlabel = 'x'
    plt.ylabel = 'y = -3x^3 - 5x^2 + 3x + 5'
    plt.plot(x, y, 'r')
    plt.plot(x, y_new, 'b')
    plt.plot(x_test, y_test, 'bo')
    plt.show()

elif type == 3:
    def fx2(x):
        return sin(x) + cos(x) ** 3


    x = linspace(-pi, pi, 1000)
    y = [fx2(el) for el in x]

    x_test = []
    print("Input number of test points:")
    cnt_points = int(input())
    cnt_points -= 1
```

```
    left = -pi
    right = pi

    step = (right - left) / cnt_points

    while True:
        x_test.append(left)
        left += step
        if left > right:
            break

    x_test0 = np.array(x_test)

    x_test = x_test0
    y_test = np.array([fx2(x) for x in x_test])

    ddc = divided_diff(x_test, y_test)[0, :]
    y_new = newton_poly(ddc, x_test, x)

    do_things_for_given_functions()

    plt.xlabel = 'x'
    plt.ylabel = 'y = sin(x) + cox(x)^3'
    plt.plot(x, y, 'r')
    plt.plot(x, y_new, 'b')
    plt.plot(x_test, y_test, 'bo')
    plt.show()
```

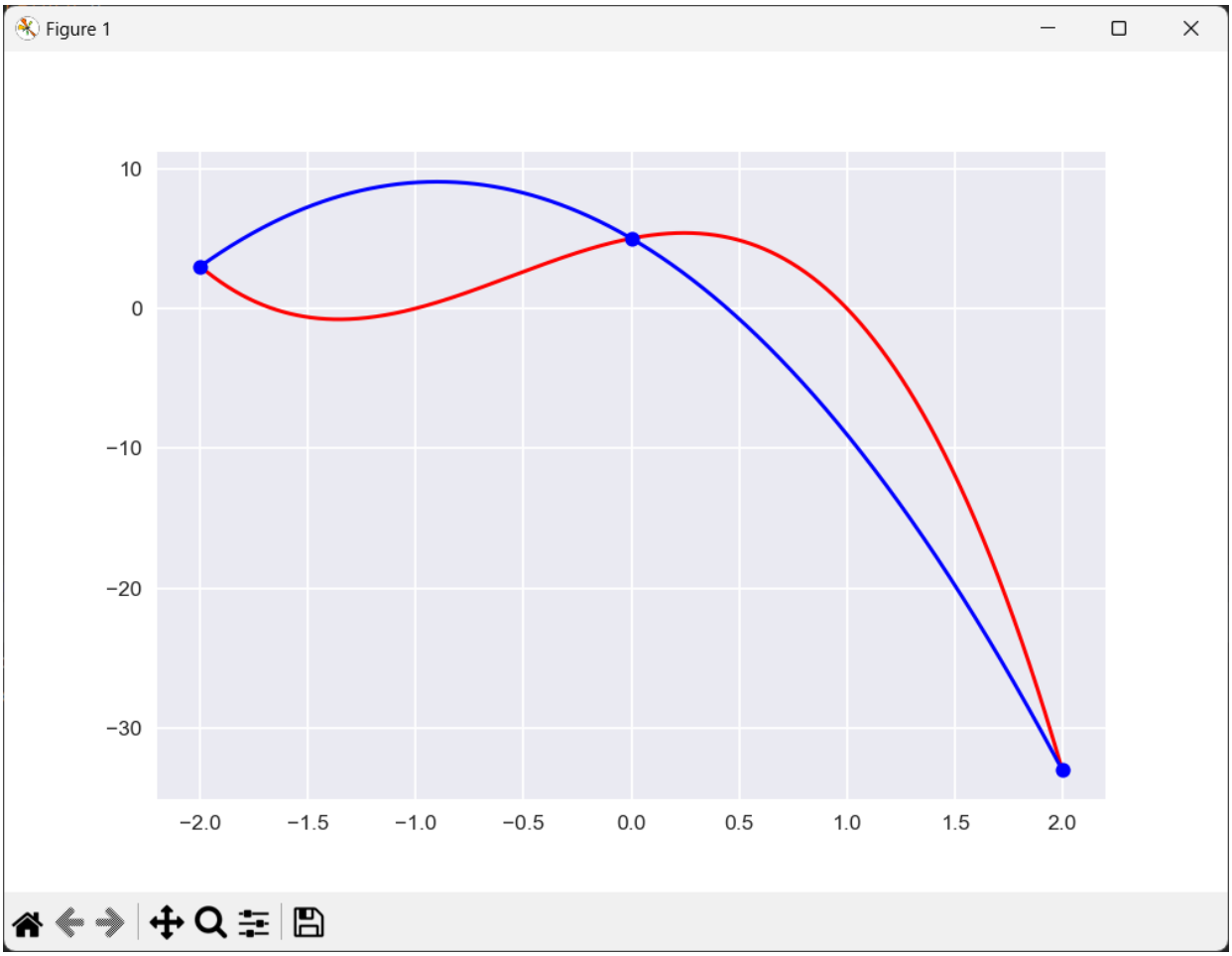
Результати роботи програми

```
1
Newton`s polynom:
12.0 - 13.0(x-x0) + 9.5(x-x0)(x-x1) - 3.6666666666666665(x-x0)(x-x1)(x-x2) + 0.7083333333333333(x-x0)(x-x1)(x-x2)(x-x3)
Newton`s polynom with given data:
12.0 - 13.0(x-3) + 9.5(x-3)(x-4) - 3.6666666666666665(x-3)(x-4)(x-5) + 0.7083333333333333(x-3)(x-4)(x-5)(x-6)
Error: 8.912973292000416e-08
```



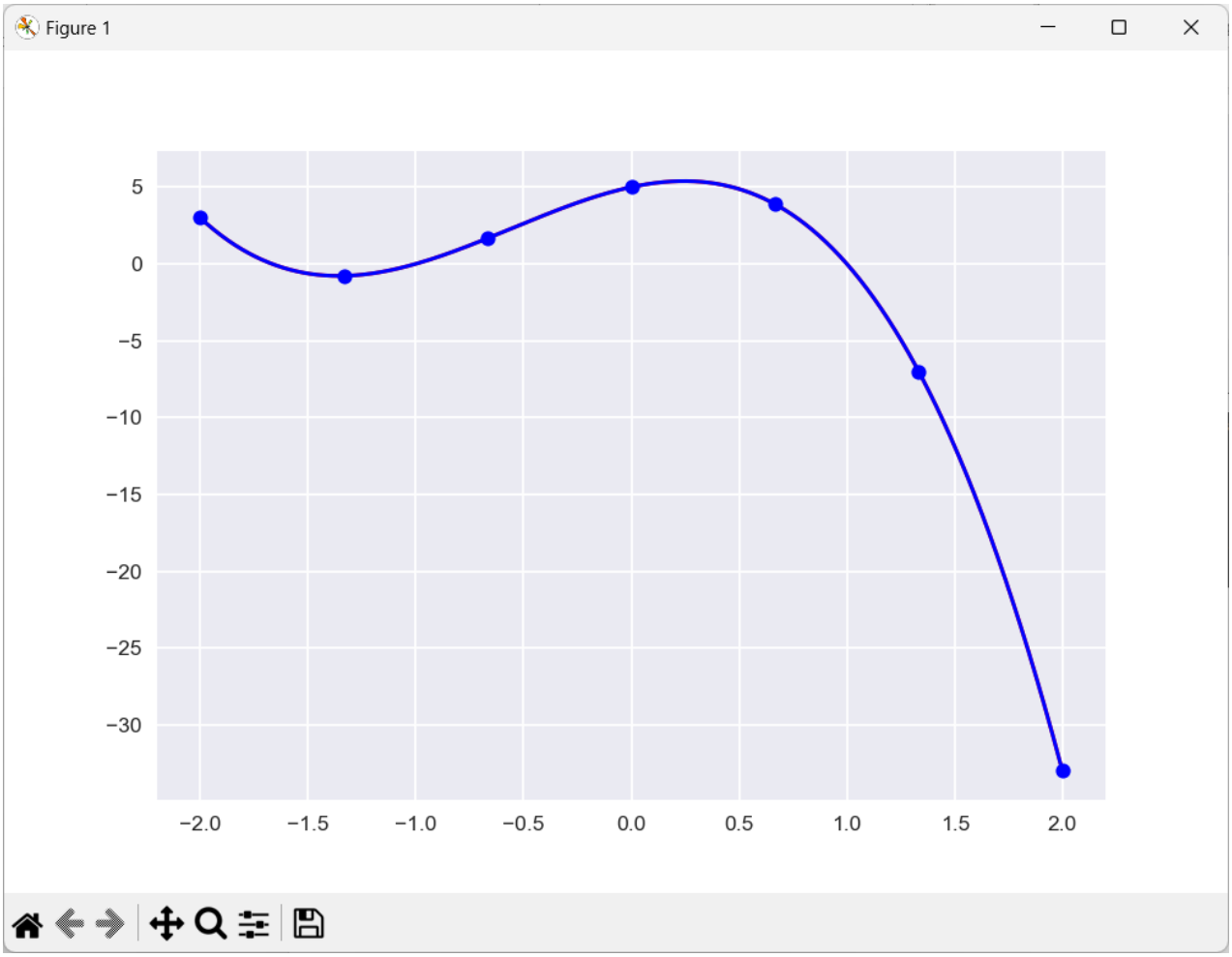
```
2
Input number of test points:
3
Newton`s polynom:
3.0 + 1.0(x-x0) - 5.0(x-x0)(x-x1)
Newton`s polynom with given data:
3.0 + 1.0(x+2.0) - 5.0(x+2.0)(x-0.0)
Error: 14.470121071980845
```

Червона лінія – y = - 3x^3 - 5x^2 + 3x + 5. Синя – за поліномом.



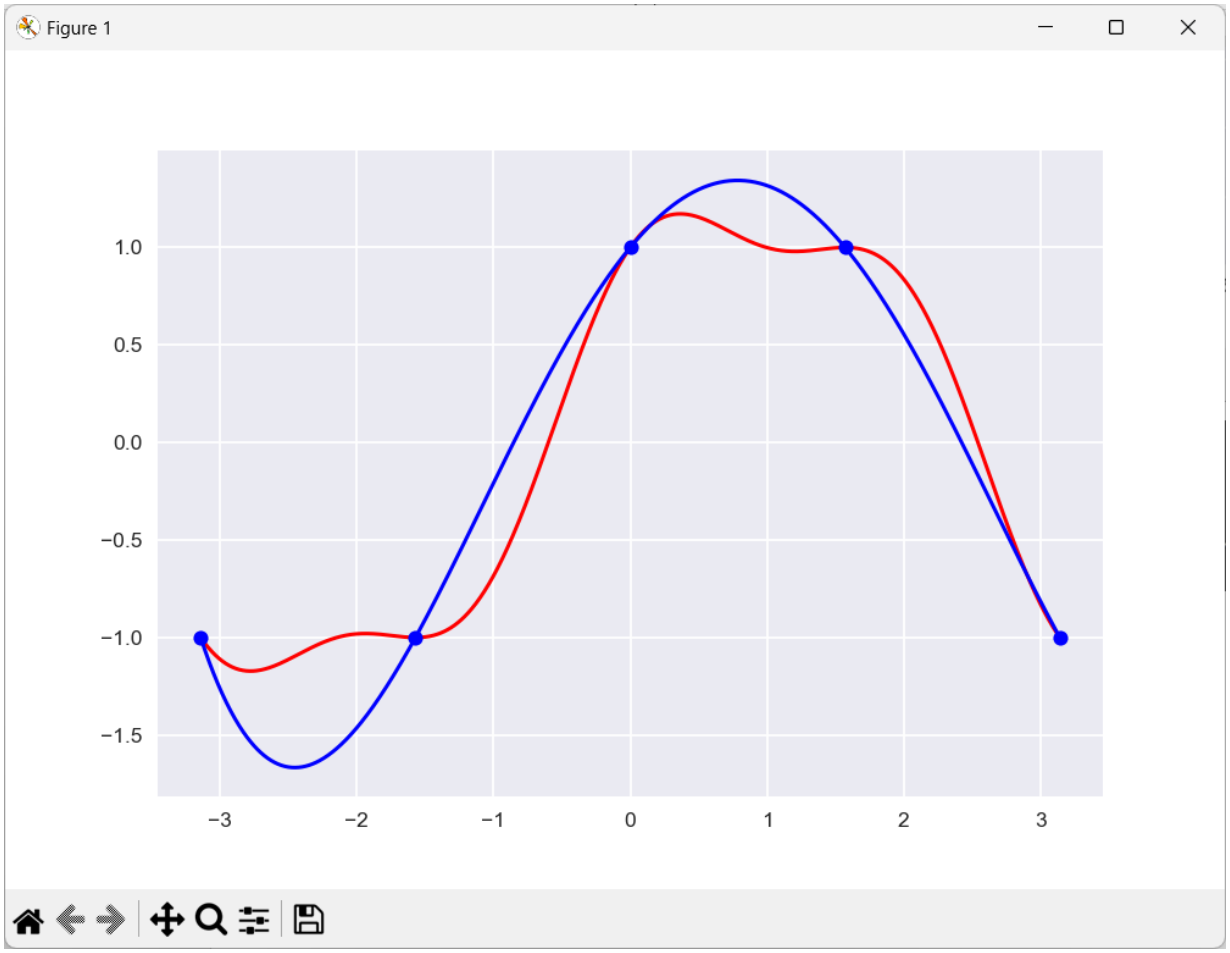
```
Please input 1 for default task, 2 for y = -3x^3 - 5x^2 + 3x + 5, 3 for y = sin(x) + cos(x)^3
2
Input number of test points:
7
Newton`s polynom:
3.0 - 5.666666666666668(x-x0) + 7.000000000000001(x-x0)(x-x1) - 3.0(x-x0)(x-x1)(x-x2) + 0.0(x-x0)(x-x1)(x-x2)(x-x3) - 9.99200722162641e-17(x-x0)(x-x1)(x-x2)(x-x3)(x-x4) +
 1.1241008124329714e-16(x-x0)(x-x1)(x-x2)(x-x3)(x-x4)(x-x5)
Newton`s polynom with given data:
3.0 - 5.666666666666668(x+2.0) + 7.000000000000001(x+2.0)(x+1.3333333333333335) - 3.0(x+2.0)(x+1.3333333333333335)(x+0.6666666666666669) + 0.0(x+2.0)(x+1.3333333333333335)(x+0
 .6666666666666669)(x+2.220446049250313e-16) - 9.99200722162641e-17(x+2.0)(x+1.3333333333333335)(x+0.6666666666666669)(x+2.220446049250313e-16)(x-0.6666666666666664) +
 1.1241008124329714e-16(x+2.0)(x+1.3333333333333335)(x+0.6666666666666669)(x+2.220446049250313e-16)(x-0.6666666666666664)(x-1.333333333333333)
Error: 3.0265724218105995e-07
```

Червона лінія – y = - 3x^3 - 5x^2 + 3x + 5. Синя – за поліномом.



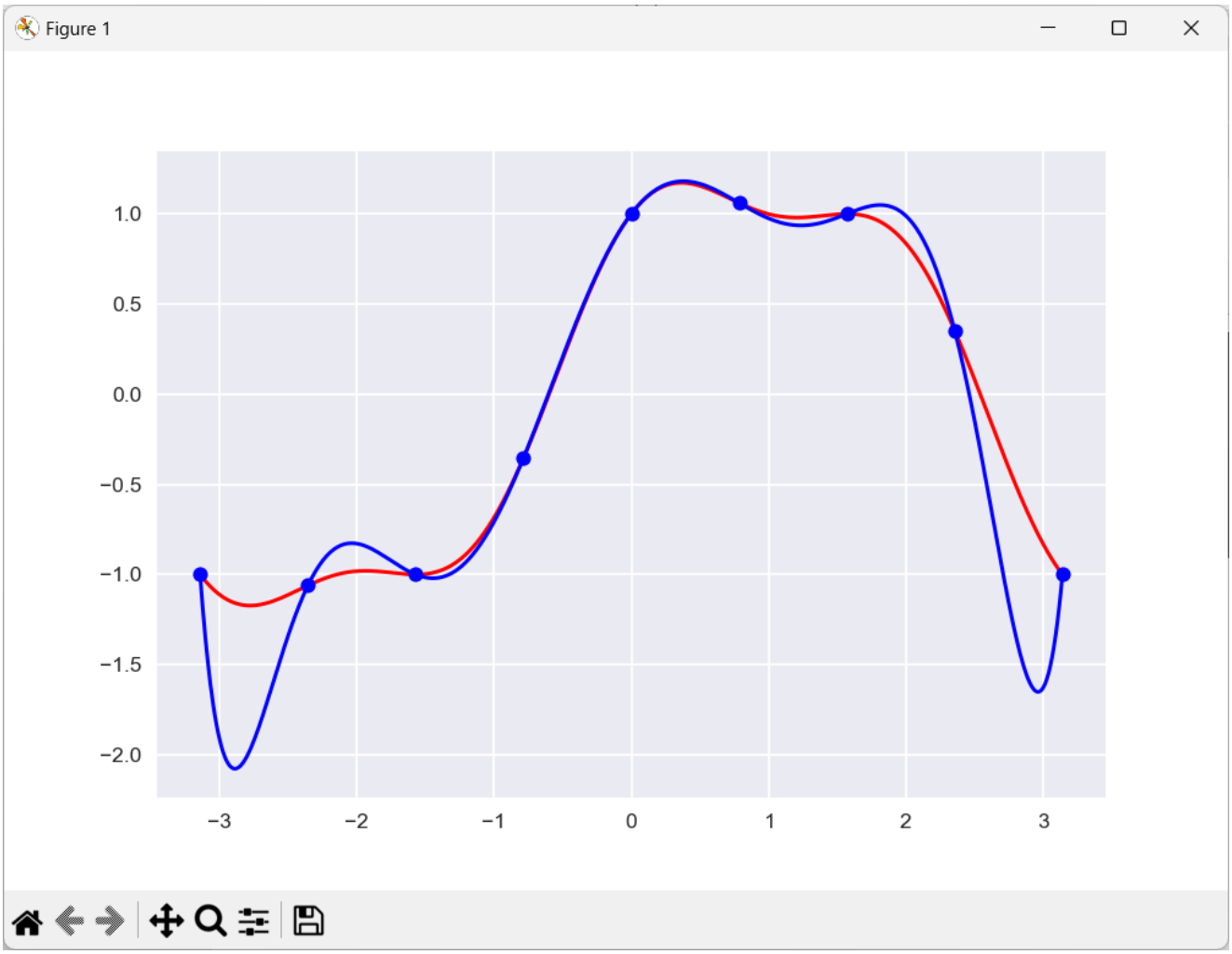
```
Please input 1 for default task, 2 for y = -3x^3 - 5x^2 + 3x + 5, 3 for y = sin(x) + cos(x)^3
3
Input number of test points:
5
Newton`s polynom:
-1.0000000000000002 + 1.4135798584282297e-16(x-x0) + 0.40528473456935105(x-x0)(x-x1) - 0.17200818364373063(x-x0)(x-x1)(x-x2) + 0.02737595267915823(x-x0)(x-x1)(x-x2)(x-x3)
Newton`s polynom with given data:
-1.0000000000000002 + 1.4135798584282297e-16(x+3.141592653589793) + 0.40528473456935105(x+3.141592653589793)(x+1.5707963267948966) - 0.17200818364373063(x+3.141592653589793)
 (x+1.5707963267948966)(x-0.0) + 0.02737595267915823(x+3.141592653589793)(x+1.5707963267948966)(x-0.0)(x-1.5707963267948966)
Error: 3.066271634927584
```

Червона лінія – y = sin(x) + cos(x)^3. Синя – за поліномом.



```
Please input 1 for default task, 2 for y = -3x^3 - 5x^2 + 3x + 5, 3 for y = sin(x) + cos(x)^3
3
Input number of test points:
9
Newton`s polynom:
-1.0000000000000002 - 0.07723492950049601(x-x0) + 0.09833856647486411(x-x0)(x-x1) + 0.15978393840921606(x-x0)(x-x1)(x-x2) - 0.037575767311803426(x-x0)(x-x1)(x-x2)(x-x3) -
 0.04958424990870739(x-x0)(x-x1)(x-x2)(x-x3)(x-x4) + 0.04184207538042238(x-x0)(x-x1)(x-x2)(x-x3)(x-x4)(x-x5) - 0.018482059809174566(x-x0)(x-x1)(x-x2)(x-x3)(x-x4)(x-x5)(x-x6) +
  0.005834326346204532(x-x0)(x-x1)(x-x2)(x-x3)(x-x4)(x-x5)(x-x6)(x-x7)
Newton`s polynom with given data:
-1.0000000000000002 - 0.07723492950049601(x+3.141592653589793) + 0.09833856647486411(x+3.141592653589793)(x+2.356194490192345) + 0.15978393840921606(x+3.141592653589793)(x+2
 .356194490192345)(x+1.5707963267948966) - 0.037575767311803426(x+3.141592653589793)(x+2.356194490192345)(x+1.5707963267948966)(x+0.7853981633974483) - 0.04958424990870739(x+3
 .141592653589793)(x+2.356194490192345)(x+1.5707963267948966)(x+0.7853981633974483)(x-0.0) + 0.04184207538042238(x+3.141592653589793)(x+2.356194490192345)(x+1
 .5707963267948966)(x+0.7853981633974483)(x-0.0)(x-0.7853981633974483) - 0.018482059809174566(x+3.141592653589793)(x+2.356194490192345)(x+1.5707963267948966)(x+0
 .7853981633974483)(x-0.0)(x-0.7853981633974483)(x-1.5707963267948966) + 0.005834326346204532(x+3.141592653589793)(x+2.356194490192345)(x+1.5707963267948966)(x+0
 .7853981633974483)(x-0.0)(x-0.7853981633974483)(x-1.5707963267948966)(x-2.356194490192345)
Error: 3.1409014926156913
```

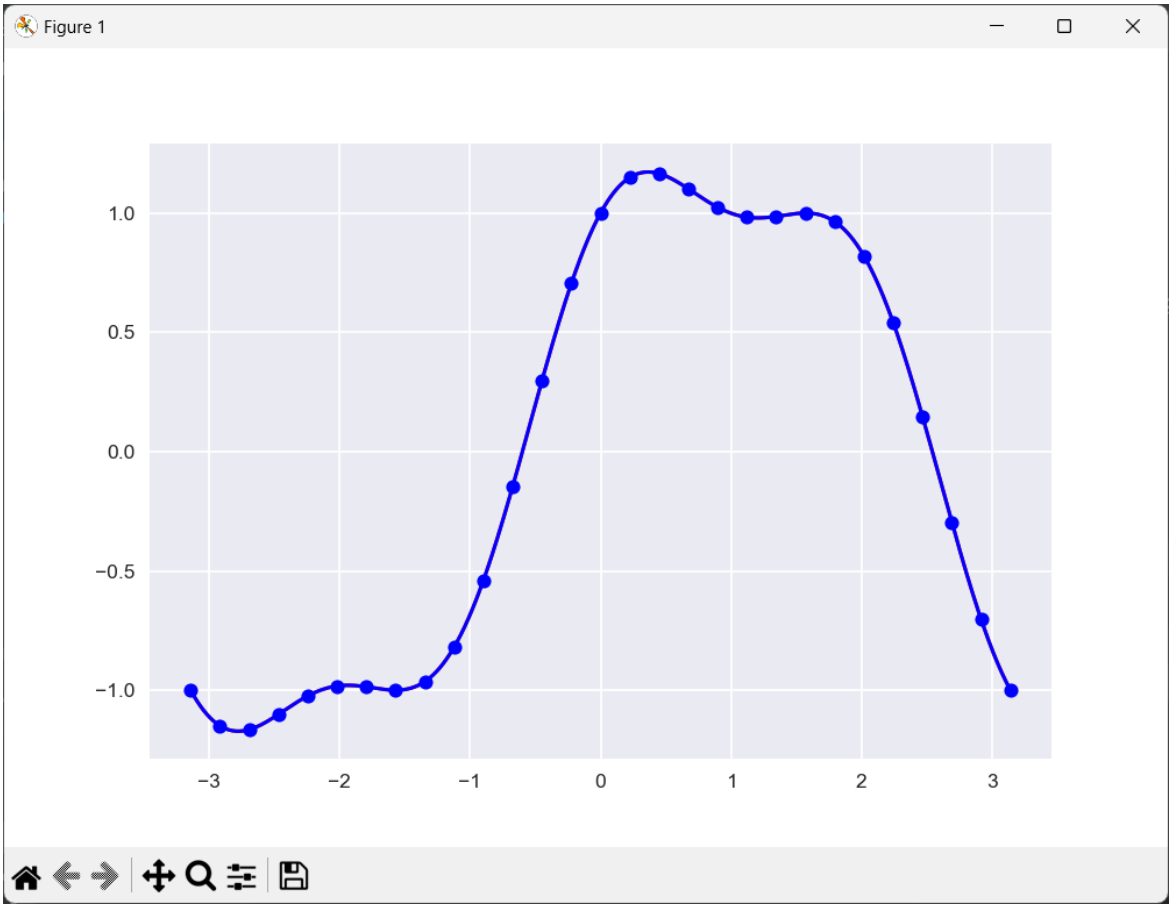
Червона лінія – y = sin(x) + cos(x)^3. Синя – за поліномом.



```
Please input 1 for default task, 2 for y = -3x^3 - 5x^2 + 3x + 5, 3 for y = sin(x) + cos(x)^3
3
Input number of test points:
29
Newton`s polynom:
-1.0000000000000002 - 0.664773117402492(x-x0) + 1.3217011696417313(x-x0)(x-x1) - 0.7846077952794202(x-
```

```
.8975979010256552)(x-1.121997376282069)(x-
.6927937030769655)(x-2.9171931783333793)
Error: 8.554648835890149e-05
```

Червона лінія – y = sin(x) + cos(x)^3. Синя – за поліномом.



Аналіз чисельних експериментів

Інтерполяційний поліном Ньютона використовують для рівновіддалених вузлів, що робить його менш гнучким.

Алгоритм знаходження інтерполяційного поліному Ньютона працює швидко і досить точно для поліноміальних функцій, з малою кількістю вхідних даних.

Алгоритм знаходження інтерполяційного поліному Ньютона працює швидко, проте не точно для тригонометричних функцій, з малою кількістю вхідних даних. Проте, з збільшенням кількості вхідних даних час буде квадратично збільшуватись, а точність буде значно збільшуватись.

Висновок

Я здобув на закріпив вміння та навички вирішення задач інтерполювання на основі побудови інтерполяційних поліномів Лагранжа та Ньютона. Створив програму, яка знаходить інтерполяційний поліном Ньютона та будує графік функції за ним.