Комп’ютерний практикум №5
Інтерполяція

Виконав:
Студент 3 курсу ФТІ
групи ФІ-92
Поночевний Назар Юрійович
Варіант 12

Завдання:
Відрізок інтерполяції розбити не менш ніж на 10 вузлів. Використовуючи аналітичне задання функції, визначене варіантом (x^2 cos x), побудувати таблицю значень функції у вузлах на відповідному відрізку інтерполяції [-π/2, π]. Побудувати за таблично заданою функцією:
- інтерполяційний поліном Pn(x) у формі Ньютона або Лагранжа;
- здійснити інтерполяцію сплайнами (другого чи третього порядку);
- побудувати графік похибки інтерполяції.

1) Реалізуємо програму

```
"""
Interpolation

"""

import math
import numpy as np
import sympy as sp
import pandas as pd
import matplotlib.pyplot as plt


# ------------ Input ------------


def f(x):
    return x**2 * math.cos(x)


A, B = -math.pi / 2, math.pi


# ------------ Code ------------
```

```
def chebyshev_nodes(n, a, b):
    nodes = []
    for k in range(1, n + 1):
        nodes.append((0.5 * (a + b)) + ((0.5 * (b - a)) * math.cos((((2
* k) - 1) * math.pi) / (2 * n))))
    return nodes


def build_lagrange_polynom(X, y):
    x = sp.Symbol('x')
    L = 0
    for i in range(len(y)):
      nominator = y[i]
      for j in range(len(y)):
        if j != i:
          nominator *= (x - X[j])
      denominator = 1
      for j in range(len(y)):
        if j != i:
          denominator *= (X[i] - X[j])
      L += (nominator / denominator)
    return sp.simplify(L)


def build_cubic_splines(x0, x, y):
    x = np.asfarray(x)
    y = np.asfarray(y)
    size = len(x)
    xdiff = np.diff(x)
    ydiff = np.diff(y)
    Li = np.empty(size)
    Li_1 = np.empty(size-1)
    z = np.empty(size)
    Li[0] = math.sqrt(2*xdiff[0])
    Li_1[0] = 0.0
    B0 = 0.0
    z[0] = B0 / Li[0]

    for i in range(1, size-1, 1):
        Li_1[i] = xdiff[i-1] / Li[i-1]
        Li[i] = math.sqrt(2*(xdiff[i-1]+xdiff[i]) - Li_1[i-1] *
Li_1[i-1])
        Bi = 6*(ydiff[i]/xdiff[i] - ydiff[i-1]/xdiff[i-1])
        z[i] = (Bi - Li_1[i-1]*z[i-1])/Li[i]

    i = size - 1
```

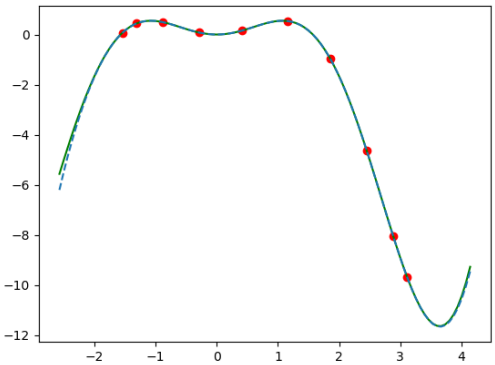
```
    Li_1[i-1] = xdiff[-1] / Li[i-1]
    Li[i] = math.sqrt(2*xdiff[-1] - Li_1[i-1] * Li_1[i-1])
    Bi = 0.0
    z[i] = (Bi - Li_1[i-1]*z[i-1])/Li[i]

    i = size-1
    z[i] = z[i] / Li[i]
    for i in range(size-2, -1, -1):
        z[i] = (z[i] - Li_1[i-1]*z[i+1])/Li[i]

    index = x.searchsorted(x0)
    np.clip(index, 1, size-1, index)
    xi1, xi0 = x[index], x[index-1]
    yi1, yi0 = y[index], y[index-1]
    zi1, zi0 = z[index], z[index-1]
    hi1 = xi1 - xi0

    f0 = zi0/(6*hi1)*(xi1-x0)**3 + \
         zi1/(6*hi1)*(x0-xi0)**3 + \
         (yi1/hi1 - zi1*hi1/6)*(x0-xi0) + \
         (yi0/hi1 - zi0*hi1/6)*(xi1-x0)
    return f0


def main():
    X = np.array(sorted(chebyshev_nodes(10, A, B)))
    print(f"Chebyshev's nodes (X):\n{X}")

    Y = np.array([f(x) for x in X])
    df = pd.DataFrame({'X': X, 'Y': Y})
    print(f"\nTable:\n{df}")

    x = sp.Symbol('x')
    L = build_lagrange_polynom(X, Y)
    print(f"\nLagrange's polynom:\n{L}")

    ls = np.linspace(A - 1, B + 1, 100)
    ls_Y_true = np.array([f(i) for i in ls])
    ls_Y_L = np.array([L.subs(x, i) for i in ls])
    r = max(np.abs(ls_Y_true - ls_Y_L))
    print(f"Max absolute error of the Lagrange's polynom: {round(r,
4)}")


    plt.scatter(X, Y, color="red")
    plt.plot(ls, ls_Y_true, color="green")
```

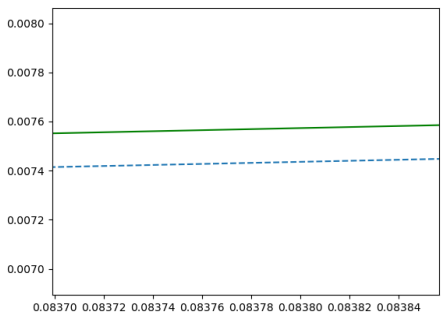
```
    plt.plot(ls, ls_Y_L, linestyle='--')
    plt.show()

    ls_Y_S = build_cubic_splines(ls, X, Y)
    r = max(np.abs(ls_Y_true - ls_Y_S))
    print(f"\nMax absolute error of the Cubic splines: {round(r, 4)}")

    plt.scatter(X, Y, color="red")
    plt.plot(ls, ls_Y_true, color="green")
    plt.plot(ls, ls_Y_S, linestyle='--')
    plt.show()


if __name__ == "__main__":
    main()
```

2) Результат

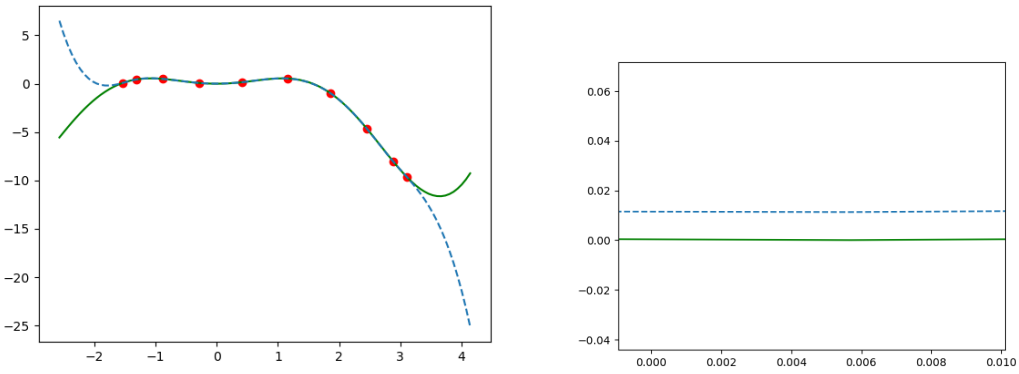
```
(base) C:\Users\Nazar\PythonWorkspace\Programming-Labs\Numerical methods\Lab5>python lab5.py
Chebyshev's nodes (X):
[-1.54178766 -1.3139865  -0.88068294 -0.28429175  0.41680814  1.15398819
  1.85508808  2.45147927  2.88478283  3.11258399]

Table:
          X         Y
0 -1.541788  0.068947
1 -1.313986  0.438540
2 -0.880683  0.493768
3 -0.284292  0.077578
4  0.416808  0.158855
5  1.153988  0.539126
6  1.855088 -0.965222
7  2.451479 -4.634562
8  2.884783 -8.049054
9  3.112584 -9.684103

Lagrange's polynom:
0.0001378886317217719*x**9 - 0.00154595268954522*x**8 - 0.000561135357636411*x**7 + 0.0429329226455653*x**6 + 0.000513525
37148408*x**5 - 0.502378360846549*x**4 + 0.000143591084302397*x**3 + 1.00141394821497*x**2 - 0.000143779398492416*x - 0.
000136108638518206
Max absolute error of the Lagrange's polynom: 0.6393

Max absolute error of the Cubic splines: 15.8524
```

Поліном Лагранжа у звичайному та збільшеному масштабі (червоним вузли, зеленим функція, синім штрихом поліном Лагранжа):





Кубічні сплайни у звичайному та збільшеному масштабі (червоним вузли, зеленим функція, синім штрихом кубічні сплайни):



3) Контрольні запитання

В чому перевага побудови полінома Ньютона у порівнянні з поліномом Лагранжа?

Поліном Ньютона легше запрограмувати, бо є можливість обчислювати розділені різниці рекурентним способом.

Яка кількість вузлів необхідна для побудови інтерполяційного полінома порядку n?

Число вузлів інтерполяційного полінома має бути на одиницю більше його ступеня.