**Лабораторна робота №5**

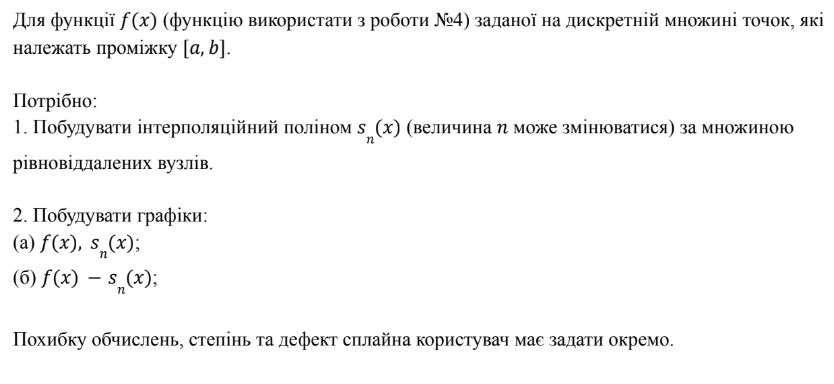
**з чисельних методів**

**Варіант №6**

**ТК-31**

**Луковського Максима Юрійовича**

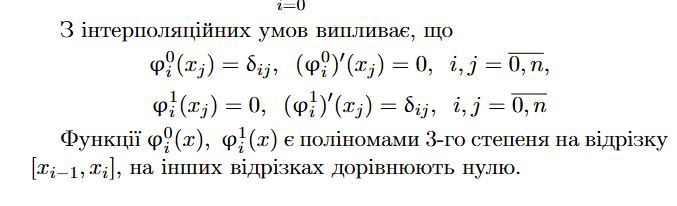
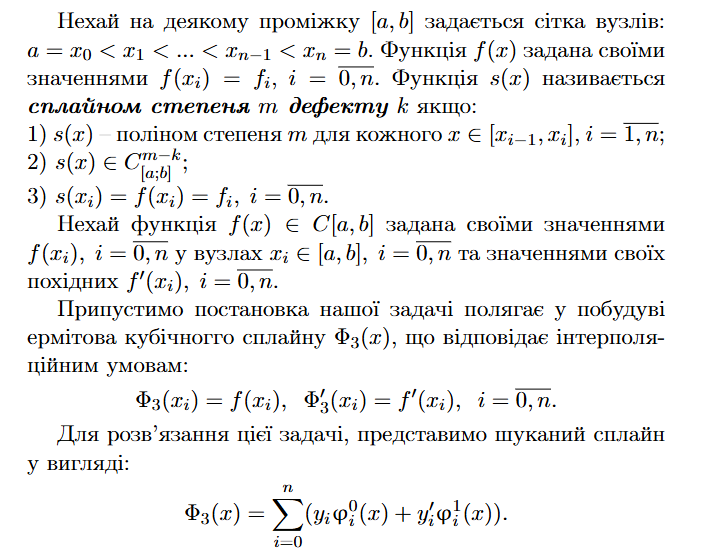
## Постановка задачі

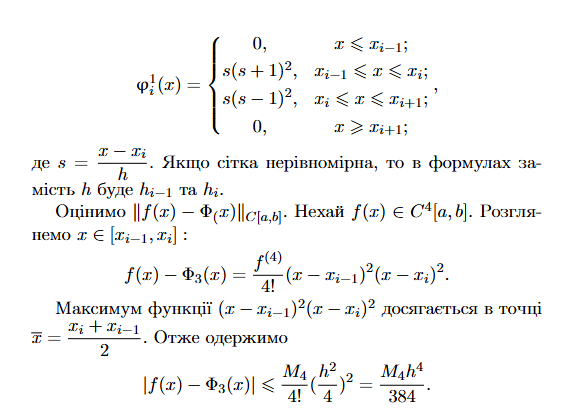
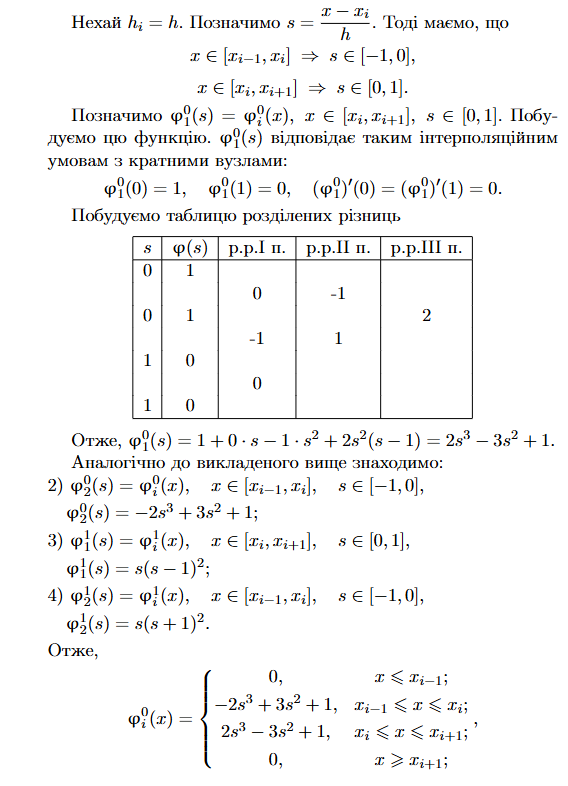




[a,b] = [-1,1]

## Теоретична частина





## Практична частина

Нехай n = 5;

Ступінь сплайна - 3

Дефект - 2

Введіть кількість вузлів n:

5

Введіть ступінь сплайна (1, 2 або 3):

3

Введіть дефект сплайна (0-{degree-1}):

2

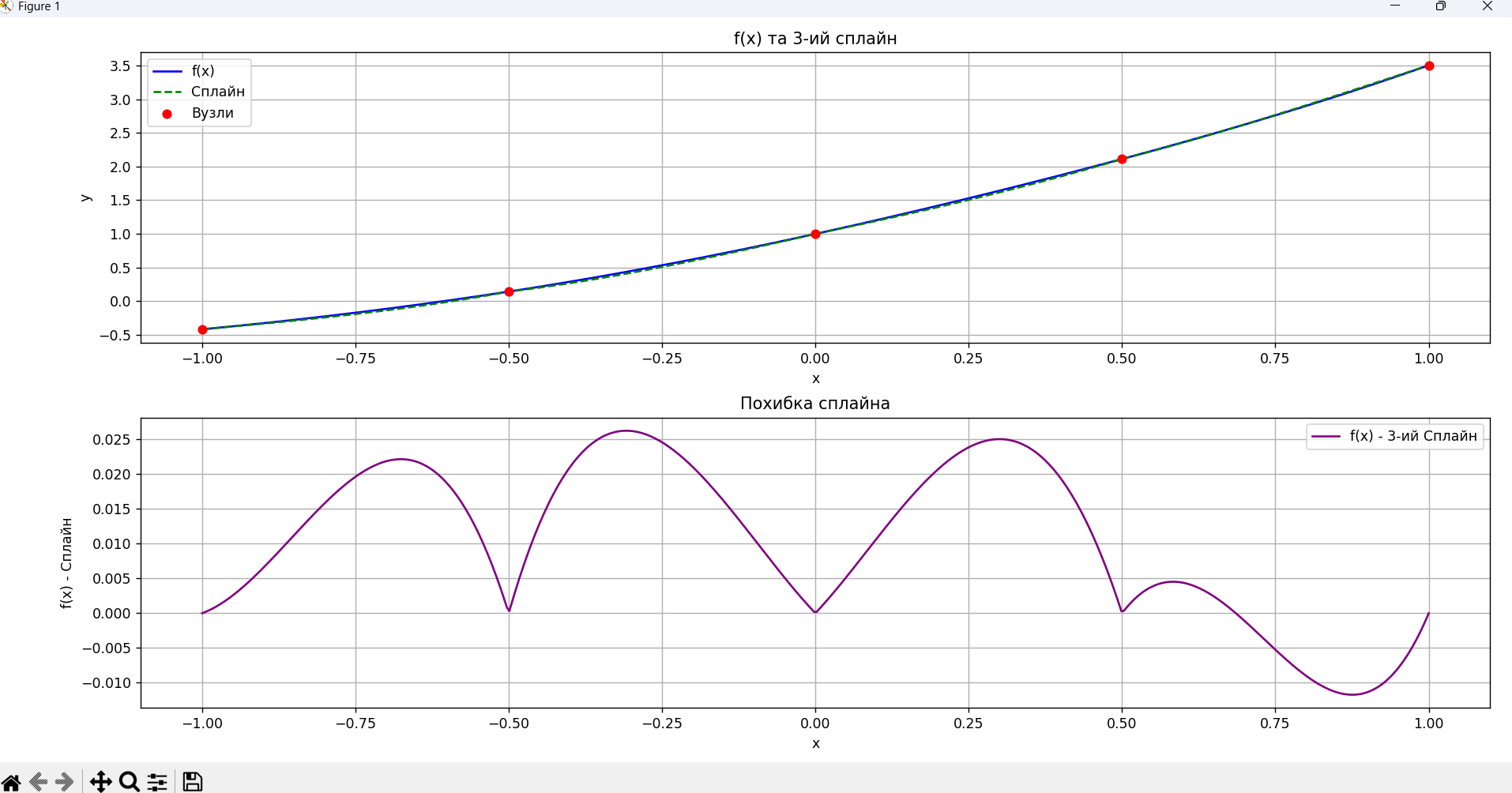
Побудовані поліноми сплайна:

S\_0(x) = -0.417519 + 0.835779\*(x - -1.000000) + 1.140098\*(x - -1.000000)^3, для x ∈ [-1.000000, -0.500000]

S\_1(x) = 0.142883 + 1.097422\*(x - -0.500000) + 1.710148\*(x - -0.500000)^2 + -0.953049\*(x - -0.500000)^3, для x ∈ [-0.500000, 0.000000]

S\_2(x) = 1.000000 + 1.905718\*(x - 0.000000) + 0.280574\*(x - 0.000000)^2 + 0.714931\*(x - 0.000000)^3, для x ∈ [0.000000, 0.500000]

S\_3(x) = 2.112369 + 2.331632\*(x - 0.500000) + 1.352970\*(x - 0.500000)^2 + -0.901980\*(x - 0.500000)^3, для x ∈ [0.500000, 1.000000]

Графіки:  


## Висновки

У цій лабораторній роботі було побудовано інтерполяційний поліном за допомогою кубічного сплайна для функції, заданої на рівномірно розподілених вузлах. Побудовані поліноми сплайна були проаналізовані за допомогою графіків, що показують як інтерпольований поліном наближається до функції та різницю між ними. Результати підтверджують високу точність апроксимації за допомогою сплайнів.

## Додатково:

import numpy as np

import matplotlib.pyplot as plt

def f(x):

return np.exp(x) + np.arctan(x)

def solve\_gauss(A, b):

n = len(A)

for i in range(n):

max\_row = i + np.argmax(np.abs(A[i:, i]))

A[[i, max\_row]] = A[[max\_row, i]]

b[[i, max\_row]] = b[[max\_row, i]]

A[i] = A[i] / A[i, i]

b[i] = b[i] / A[i, i]

for j in range(i + 1, n):

b[j] -= b[i] \* A[j, i]

A[j] -= A[i] \* A[j, i]

x = np.zeros(n)

for i in range(n - 1, -1, -1):

x[i] = b[i] - np.sum(A[i, i + 1:] \* x[i + 1:])

return x

def cubic\_spline(x\_nodes, y\_nodes, degree=3, defect=0):

n = len(x\_nodes) - 1

h = np.diff(x\_nodes)

if degree > 3 or degree < 1:

raise ValueError("Degree must be 1, 2, or 3")

if defect < 0 or defect >= degree:

raise ValueError(f"Defect must be between 0 and {degree-1}")

if degree == 3:

alpha = np.zeros(n - 1)

for i in range(1, n):

alpha[i - 1] = (3 / h[i] \* (y\_nodes[i + 1] - y\_nodes[i]) -

3 / h[i - 1] \* (y\_nodes[i] - y\_nodes[i - 1]))

A = np.zeros((n - 1, n - 1))

for i in range(n - 1):

if i > 0:

A[i, i - 1] = h[i]

A[i, i] = 2 \* (h[i] + h[i + 1] if i < n - 2 else h[i])

if i < n - 2:

A[i, i + 1] = h[i + 1]

c = np.zeros(n + 1)

if n > 1:

c[1:n] = solve\_gauss(A, alpha)

b = np.zeros(n)

d = np.zeros(n)

for i in range(n):

b[i] = ((y\_nodes[i + 1] - y\_nodes[i]) / h[i] -

h[i] \* (c[i + 1] + 2 \* c[i]) / 3)

d[i] = (c[i + 1] - c[i]) / (3 \* h[i])

return y\_nodes[:-1], b, c[:-1], d, x\_nodes

elif degree == 2:

b = (y\_nodes[1:] - y\_nodes[:-1]) / (x\_nodes[1:] - x\_nodes[:-1])

c = np.zeros\_like(b)

d = np.zeros\_like(b)

return y\_nodes[:-1], b, c, d, x\_nodes

elif degree == 1:

b = np.zeros\_like(y\_nodes[:-1])

c = np.zeros\_like(b)

d = np.zeros\_like(b)

return y\_nodes[:-1], b, c, d, x\_nodes

def evaluate\_spline(x, coeffs):

a, b, c, d, x\_nodes = coeffs

for i in range(len(x\_nodes) - 1):

if x\_nodes[i] <= x <= x\_nodes[i + 1]:

dx = x - x\_nodes[i]

return a[i] + b[i] \* dx + c[i] \* dx\*\*2 + d[i] \* dx\*\*3

return None

def spline\_to\_string(coeffs):

a, b, c, d, x\_nodes = coeffs

polynomials = []

for i in range(len(a)):

poly = f"S\_{i}(x) = {a[i]:.6f}"

if b[i] != 0:

poly += f" + {b[i]:.6f}\*(x - {x\_nodes[i]:.6f})"

if c[i] != 0:

poly += f" + {c[i]:.6f}\*(x - {x\_nodes[i]:.6f})^2"

if d[i] != 0:

poly += f" + {d[i]:.6f}\*(x - {x\_nodes[i]:.6f})^3"

poly += f", для x ∈ [{x\_nodes[i]:.6f}, {x\_nodes[i+1]:.6f}]"

polynomials.append(poly)

return polynomials

a, b = -1, 1

print("Введіть кількість вузлів n:")

n = int(input())

print("Введіть степінь сплайна (1, 2 або 3):")

degree = int(input())

print("Введіть дефект сплайна (0-{degree-1}):")

defect = int(input())

x\_nodes = np.linspace(a, b, n)

y\_nodes = f(x\_nodes)

coeffs = cubic\_spline(x\_nodes, y\_nodes, degree, defect)

polynomials = spline\_to\_string(coeffs)

print("\nПобудовані поліноми сплайна:")

for poly in polynomials:

print(poly)

x\_values = np.linspace(a, b, 500)

f\_values = f(x\_values)

spline\_values = [evaluate\_spline(x, coeffs) for x in x\_values]

plt.figure(figsize=(10, 8))

plt.subplot(2, 1, 1)

plt.plot(x\_values, f\_values, label='f(x)', color='blue')

plt.plot(x\_values, spline\_values, label='Сплайн', linestyle='--', color='green')

plt.scatter(x\_nodes, y\_nodes, color='red', zorder=5, label='Вузли')

plt.title(f'f(x) та {degree}-ий сплайн')

plt.xlabel('x')

plt.ylabel('y')

plt.grid(True)

plt.legend()

plt.subplot(2, 1, 2)

error\_values = f\_values - np.array(spline\_values)

plt.plot(x\_values, error\_values, label=f'f(x) - {degree}-ий Сплайн', color='purple')

plt.title('Похибка сплайна')

plt.xlabel('x')

plt.ylabel('f(x) - Сплайн')

plt.grid(True)

plt.legend()

plt.tight\_layout()

plt.show()

print(f"\nМаксимальна похибка: {np.max(np.abs(error\_values)):.6f}")