

# Metody Numeryczne - Zadanie VII

Mateusz Kamiński

November 2025

## 1 Wprowadzenie

Celem ćwiczenia było zaimplementowanie metody interpolacji Lagrange'a w celu wyznaczenia wielomianu interpolacyjnego dla zadanych węzłów.

## 2 Opis

### 2.1 Wzór bazowy

Wielomian interpolacyjny w postaci Lagrange'a ma postać:

$$P(x) = \sum_{j=1}^n \ell_j(x) f_j,$$

gdzie  $\ell_j(x)$  są wielomianami bazowymi Lagrange'a:

$$\ell_j(x) = \prod_{\substack{k=1 \\ k \neq j}}^n \frac{x - x_k}{x_j - x_k}.$$

W szczególności:

$$a_0 = P(0) = \sum_{j=1}^n \ell_j(0) f_j.$$

### 2.2 Rekurencja do wyznaczania kolejnych współczynników

Po odjęciu wyrazu wolnego otrzymujemy funkcję:

$$\frac{P(x) - a_0}{x} = a_1 + a_2 x + \cdots + a_n x^{n-1},$$

która sama jest wielomianem stopnia o 1 mniejszego. Jej wartości w punktach węzłowych wynoszą:

$$\frac{f_i - a_0}{x_i}.$$

Korzystając ponownie z interpolacji Lagrange'a, możemy wyznaczyć kolejny współczynnik  $a_1$ :

$$a_1 = \sum_{j=0}^n \ell_j(0) \left( \frac{f_j - a_0}{x_j} \right).$$

Powtarzając ten proces, otrzymujemy rekurencję dla wszystkich współczynników:

$$a_k = \sum_{j=0}^n \ell_j(0) f_j^{(k)},$$

gdzie:

$$f_j^{(0)} = f_j, \quad f_j^{(k)} = \frac{f_j^{(k-1)} - a_{k-1}}{x_j}.$$

Metoda działa w czasie  $O(n^2)$  i pozwala wyznaczyć wszystkie współczynniki wielomianu bez rozwiązywania układu równań liniowych.

### 2.3 Schemat Hornera

Po wyznaczeniu współczynników wartości wielomianu obliczano za pomocą schematu Hornera:

$$P(x) = (\dots((a_{n-1}x + a_{n-2})x + a_{n-3})x + \dots)x + a_0.$$

## 3 Implementacja

```
import numpy as np
from numpy.typing import NDArray
import matplotlib.pyplot as plt

vector = NDArray[np.float64]
matrix = NDArray[np.float64]
num = np.float64

def l_j(x_val: num, x: vector, j: int) -> num:
    n = len(x)
    numerator: num = np.float64(1.0)
    denominator: num = np.float64(1.0)

    # 0...j-1
    for k in range(0, j):
```

```

        numerator *= x_val - x[k]
        denominator *= x[j] - x[k]

    # j+1...n-1
    for k in range(j + 1, n):
        numerator *= x_val - x[k]
        denominator *= x[j] - x[k]

    return numerator / denominator

def coefficients(x: vector, f: vector) -> matrix:
    n = len(x)

    l_j = np.zeros(n, dtype=num)
    for j in range(n):
        l_j[j] = l_j(0, x, j)

    # współczynniki
    a = np.zeros(n, dtype=num)
    f_k = f.copy()

    for k in range(n):
        a[k] = np.sum(l_j * f_k)

        for j in range(n):
            f_k[j] = (f_k[j] - a[k]) / x[j]

    return a

def P(x: num, a: vector) -> num:
    # Wyliczenie wielomianu schematem hornera
    val = 0.0
    for coeff in reversed(a):
        val = val * x + coeff

    return val

def main():
    x: vector = np.array([-1.00, -0.75, -0.50, -0.25, 0.25, 0.50, 0.75,
                          1.00], dtype=num)
    f: vector = np.array([
        6.00000000000000,
        3.04034423828125,
        1.74218750000000,
        1.26361083984375,
        0.75982666015625,
        0.63281250000000,
        0.85809326171875,
        2.00000000000000
    ], dtype=num)

```

```

n = len(x)
coeff = coefficients(x, f)
print(coeff)

x_plot = np.linspace(-2, 1.25, 1000)
y_plot = np.array([P(x, coeff) for x in x_plot])

plt.plot(x_plot, y_plot, label="Wielomian interpolacyjny", color="blue")
plt.scatter(x, f, color="red", label="Punkty interpolacji", zorder=5)

plt.xlabel("x")
plt.ylabel("f(x)")
plt.title("Interpolacja Lagrange'a")
plt.grid(True)
plt.legend()
plt.xlim(-2, 1.25)
plt.show()

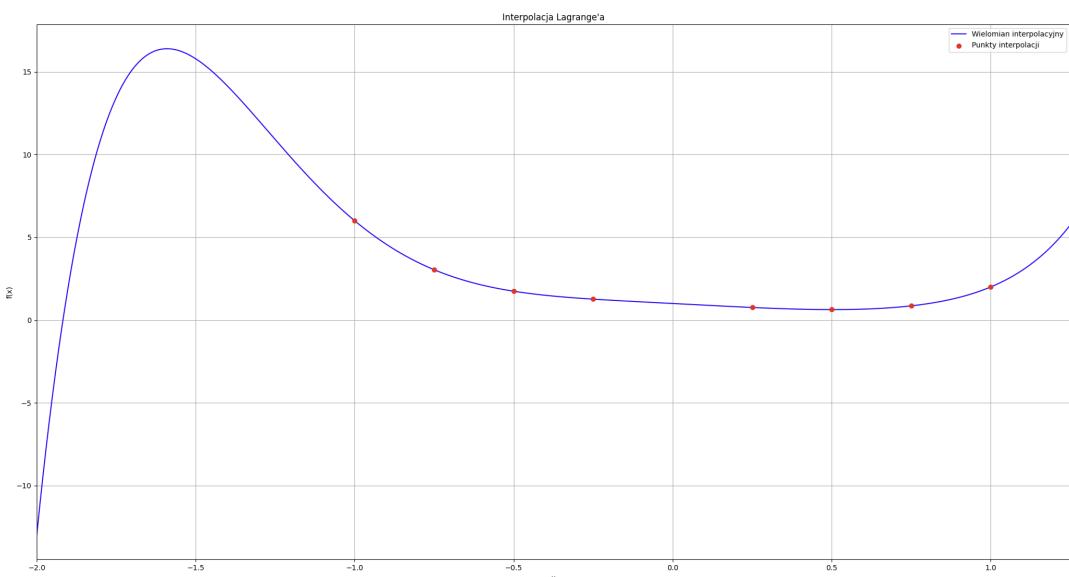
if __name__ == "__main__":
    main()

```

## 4 Wyniki

Wektor współczynników **a** wielomianu interpolacyjnego:

$$\begin{aligned}
 a_0 &= 1.0000, \\
 a_1 &= -1.0000, \\
 a_2 &= -0.0000, \\
 a_3 &= 0.0000, \\
 a_4 &= 3.0000, \\
 a_5 &= -2.0000, \\
 a_6 &= 0.0000, \\
 a_7 &= 1.0000.
 \end{aligned}$$



Rysunek 1: Wyniki interpolacji Lagrange'a.