Zadanie Numeryczne 08

Mateusz Wojtyna

1. Wstęp

Należało znaleźć wartości funkcji

$$f(x) = \frac{1}{1+5x^2}$$

w punktach $-\frac{7}{8}$, $-\frac{5}{8}$, $-\frac{3}{8}$, $-\frac{1}{8}$, $\frac{1}{8}$, $\frac{3}{8}$, $\frac{5}{8}$, $\frac{7}{8}$, a następnie skonstruować wielomian interpolacyjny Lagrange'a oraz narysować wykres tego wielomianu. Znaleziono również współczynniki tego wielomianu.

2. Opis

2.1. Interpolacja Lagrange'a

Wielomian interpolacyjny Lagrange'a dla funkcji o n węzłach dany jest następującym wzorem:

$$P(x) = \sum_{i=1}^{n} l_i(x) y_i$$

gdzie $l_i(x)$ jest wielomianem bazowym Lagrange'a:

$$l_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

2.2. Wyznaczenie współczynników wielomianu interpolacyjnego

Możemy wyznaczyć współczynnik a_0 obliczając:

$$a_0 = P(0) = \sum_{i=1}^{n} l_i(0) y_i$$

Zajmie to $\mathcal{O}(n^2)$ czasu, ponieważ dla każdego i musimy policzyć jeszcze $l_i(0)$ co zajmuje $\mathcal{O}(n)$ czasu. Po wyliczeniu każdego $l_i(0)$ możemy zapisać go w pamięci do wykorzystania w następnych współczynnikach.

Współczynniki $a_{k>0}$ można obliczyć z następującego wzoru:

$$a_k = \sum_{i=1}^{n} l_i(0) y_i^{(k)}$$

gdzie:

$$y_i^{(k)} = \begin{cases} y_i & k = 0 \\ \frac{y_i^{(k-1)} - a_{k-1}}{x_i} & k \in \{1, \ldots, n-1\} \end{cases}$$

Wyliczamy je w czasie $\mathcal{O}(n)$, ponieważ mamy już zapamiętany $l_i(0)$, a $y_i^{(k)}$ możemy wyliczyć w czasie stałym dla każdego i.

3. Kod

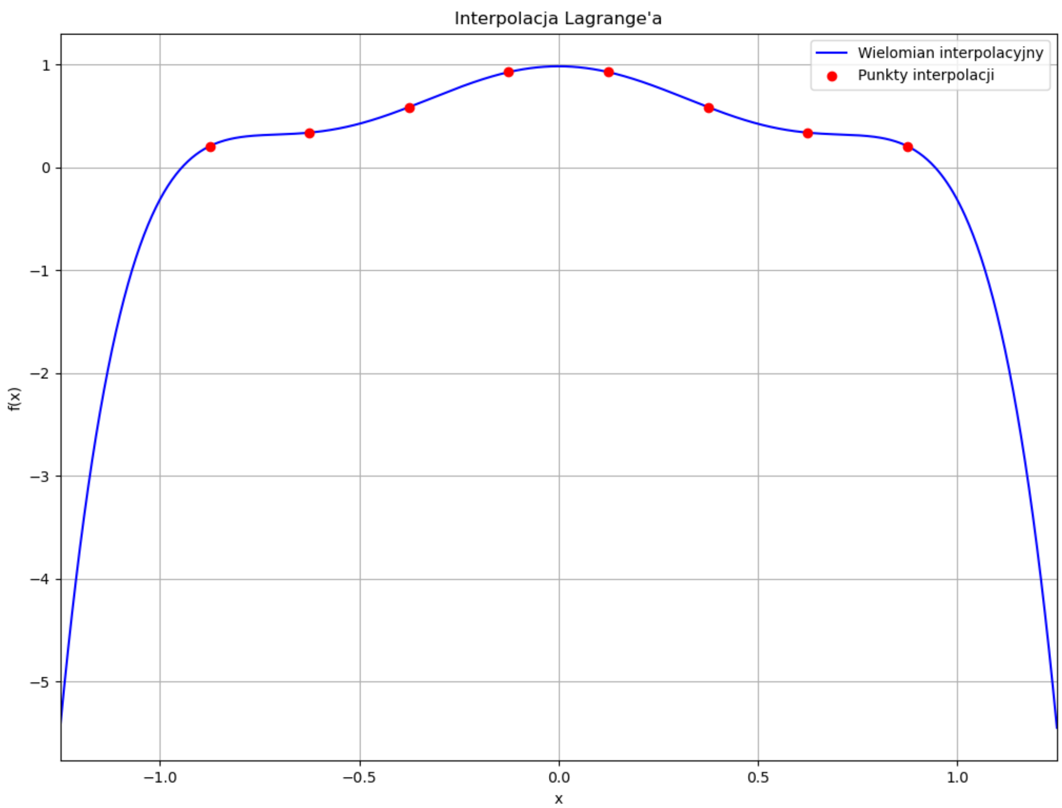
Program napisano w Pythonie z użyciem pakietu *NumPy* oraz *Matplotlib*.

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

# Dla przejrzystości kodu
vector = NDArray[np.float64]


def lagrange_basis(j: int, x: float, nodes: vector) -> float:
    N = len(nodes)
    nominator = 1.0
    denominator = 1.0

    for i in range(0, j):
        nominator *= x - nodes[i]
        denominator *= nodes[j] - nodes[i]
    for i in range(j + 1, N):
        nominator *= x - nodes[i]
        denominator *= nodes[j] - nodes[i]

    return nominator / denominator


def lagrange_polynomial(nodes: vector, values: vector) -> vector:
    """
    Zwraca współczynniki wielomianu interpolacyjnego Lagrange'a
    """
    N = len(nodes)
    a = np.zeros(N, dtype=np.float64)
    l = np.zeros(N, dtype=np.float64)
    values = values.copy()

    # k = 0
    for j in range(N):
        l[j] = lagrange_basis(j, 0, nodes)
```

```python
        a[0] += l[j] * values[j]

    # k > 0
    for k in range(1, N):
        for j in range(N):
            values[j] = (values[j] - a[k - 1]) / nodes[j]
            a[k] += l[j] * values[j]

    return a


def f(x: float):
    return 1 / (1 + 5 * (x**2))


def P(x: float, a: vector):
    N = len(a)

    y = 0
    tmp = 1
    for i in range(N):
        y += a[i] * tmp
        tmp *= x

    return y


def main():
    np.set_printoptions(suppress=True)

    nodes = np.array([-7 / 8, -5 / 8, -3 / 8, -1 / 8, 1 / 8, 3 / 8, 5 / 8, 7 /
         8])
    values = np.array([f(x) for x in nodes])

    a: vector = lagrange_polynomial(nodes, values)
    print(a)

    x_plot = np.linspace(-1.25, 1.25, 1000)
    y_plot = np.array([P(x, a) for x in x_plot])
    plt.plot(x_plot, y_plot, label="Wielomian interpolacyjny", color="blue")
    plt.scatter(nodes, values, 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(-1.25, 1.25)
    plt.show()


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

4. Wyniki

$$
\begin{aligned}
a_0 &= 0.98431217,\\
a_1 &= 0,\\
a_2 &= -3.74533114,\\
a_3 &= 0,\\
a_4 &= 7.22153333,\\
a_5 &= 0,\\
a_6 &= -4.77456749,\\
a_7 &= 0.
\end{aligned}
$$



Rysunek 1: Wielomian interpolacyjny funkcji f