Metody Numeryczne - Zadanie XIX

Mateusz Kamiński

December 2025

1 Wstęp

Celem zadania było wyznaczenie takiej wartości parametru α, dla której pochodna wielomianu interpolacyjnego Lagrange'a $l(x, \alpha)$ w punkcie $x = 7$ przyjmuje wartość zero.

2 Opis

2.1 Liniowość

Wielomian interpolacyjny oparty na węzłach (x_i, y_i) można przedstawić jako sumę funkcji bazowych Lagrange'a:

$$l(x, \alpha) = \sum_{i=0}^{n} y_i L_{i(x)}$$

W naszym przypadku tylko jedna wartość y_i zależy od α (dla $x = 4$). Możemy więc zapisać:

$$l(x, \alpha) = \alpha L_3(x) + C(x)$$

gdzie $C(x)$ to suma składników niezależnych od α. Ponieważ pochodna jest operatorem liniowym, funkcja

$$p(\alpha) = \frac{\partial l(x, \alpha)}{\partial x} \Big|_{x=7}$$

jest funkcją liniową postaci:

$$p(\alpha) = A\alpha + B$$

2.2 Algorytm rozwiązania

Aby znaleźć miejsce zerowe $p(\alpha) = 0$, wyznaczono parametry poprzez dwukrotne obliczenie pochodnej wielomianu dla różnych wartości α:

1. Obliczono $p(0)$ (co odpowiada wyrazowi wolnemu B).
2. Obliczono $p(1)$ (co odpowiada sumie $A + B$).
3. Wyznaczono α_0 ze wzoru:

$$p(\alpha_0) = 0 = A\alpha_0 + B \quad \rightarrow \quad \alpha_0 = -\frac{B}{p(1) - p(0)}$$

3 Implementacja

Python

```python
import numpy as np
from numpy.typing import NDArray

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

def p_alpha(alpha: num) -> num:
    x = np.array([1, 2, 3, 4, 5, 6, 7], dtype=float)
    y = np.array([1, 0, 1, alpha, 1, 0, 1], dtype=float)

    a = coefficients(x, y)
    return derivative_at_point_x(a, 7)

def derivative_at_point_x(a: vector , x: num) -> num:
    total = 0.0

    # Schemat Hornera
    # b_k = (k + 1) * a_k+1
    for k in range(len(a) - 1, 0, -1):
        total = total * x + k * a[k]

    return total

def l_j(x_val: num, x: vector, j: int) -> num:
    n = len(x)
```

```
26      numerator: num = np.float64(1.0)
27      denominator: num = np.float64(1.0)
28
29      # 0...j-1
30      for k in range(0, j):
31          numerator *= x_val - x[k]
32          denominator *= x[j] - x[k]
33
34      # j+1...n-1
35      for k in range(j + 1, n):
36          numerator *= x_val - x[k]
37          denominator *= x[j] - x[k]
38
39      return numerator / denominator
40
41  def coefficients(x: vector, f: vector) -> vector:
42      n = len(x)
43
44      lj = np.zeros(n, dtype=num)
45      for j in range(n):
46          lj[j] = l_j(0, x, j)
47
48      # współczynniki
49      a = np.zeros(n, dtype=num)
50      f_k = f.copy()
51
52      for k in range(n):
53          a[k] = np.sum(lj * f_k)
54
55          for j in range(n):
56              f_k[j] = (f_k[j] - a[k]) / x[j]
57
58      return a
59
60  def main():
61      # Funkcja spełnia zależność liniową
62      # p_alfa = A*alfa + B
63
64      p0 = p_alpha(0.0) # = B
```

```
65      p1 = p_alpha(1.0) # = A + B
66
67      A = p1 - p0
68      B = p0
69
70      # chcemy żeby funkcja p_alfa, czyli pochodna po Lagrangu się
        zerowała, więc
71      # A * alfa + B = 0
72      # alfa = -B / A
73
74      alpha0 = -B / A
75
76      print("Miejsce zerowe p(alfa) = ", alpha0, "")
77
78  if __name__ == "__main__":
79      main()
```

4 Wyniki

4.1 Rozwiązanie

Wyznaczono wartość parametru α, dla której nachylenie wielomianu w prawym krańcu przedziału zanika:

$$\alpha_0 = 2.080000001587457$$