Algorytmy Numeryczne 1.1 - Interpolacja Newtona

Przemyslaw Sawoniuk, Jeremiasz Olech

26.11.2025

1 Podstawy teoretyczne

Dla $n+1$ punktow $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, gdzie x_i sa rozne, wielomian interpolacyjny w **postaci Newtona** ma postac:

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)\cdots(x - x_{n-1}),$$

gdzie:

- $a_k = f[x_0, x_1, \ldots, x_k]$ to **ilorazy roznicowe** rzedu k,
- Ilorazy obliczane sa rekurencyjnie:

$$f[x_i, \ldots, x_j] = \frac{f[x_{i+1}, \ldots, x_j] - f[x_i, \ldots, x_{j-1}]}{x_j - x_i}.$$

2 Kod w Pythonie

2.1 Pelny kod

```
import sys
from validation import validate_and_parse

def main():
    print(
        "Podaj dane w nastepujacym formacie:\n"
        "n\n"
        "x0 x1 ... xn\n"
        "y0 y1 ... yn\n"
        "t1 t2 ... (opcjonalnie kolejne linie z t)\n\n"
        "Po zakonczeniu wpisywania danych nacisnij:\n"
        " - Ctrl+D  (Linux/Mac)\n"
        " - Ctrl+Z+Enter  (Windows)\n"
    )
    data = sys.stdin.read().splitlines()
    try:
        n, x, y, t_values = validate_and_parse(data)
    except ValueError as e:
        print("Blad:", e)
        return

    m = n + 1
    coeffs = y.copy()

    for i in range(1, m):
        for j in range(m - 1, i - 1, -1):
            coeffs[j] = (coeffs[j] - coeffs[j - 1]) / (x[j] - x[j - i])

    results = []
    for t in t_values:
        result = coeffs[-1]
```

```
        for i in range(m - 2, -1, -1):
            result = coeffs[i] + (t - x[i]) * result
        results.append(result)

    for res in results:
        print(res)

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

Listing 1: Interpolacja Newtona z obsluga bledow

2.2 Modul walidacji

```
def validate_and_parse(data):
    if not data:
        raise ValueError("Brak danych wejsciowych.")

    # n
    try:
        n = int(data[0].strip())
    except ValueError:
        raise ValueError("Pierwsza linia musi zawierac liczbe calkowita n.")

    if n < 1:
        raise ValueError("n musi byc >= 1.")

    if len(data) < 3:
        raise ValueError("Zbyt malo linii danych - oczekiwane: n, x, y.")

    # x
    try:
        x = list(map(float, data[1].split()))
    except ValueError:
        raise ValueError("Linia 2 musi zawierac wartosci liczbowe x.")

    if len(x) != n + 1:
        raise ValueError(f"Dla n={n} nalezy podac dokladnie {n+1} wartosci x (
    podano {len(x)}).")

    if len(set(x)) != len(x):
        raise ValueError("Wartosci x musza byc unikalne.")

    # y
    try:
        y = list(map(float, data[2].split()))
    except ValueError:
        raise ValueError("Linia 3 musi zawierac wartosci liczbowe y.")

    if len(y) != n + 1:
        raise ValueError(f"Dla n={n} nalezy podac dokladnie {n+1} wartosci y (
    podano {len(y)}).")

    # t
    t_values = []
    for line in data[3:]:
        try:
            t_values.extend(map(float, line.split()))
        except ValueError:
            raise ValueError("W liniach z t musza znajdowac sie liczby.")

    if not t_values:
        raise ValueError("Brak punktow t do obliczenia interpolacji.")
```

```
48
49      return n, x, y, t_values
```

Listing 2: validation.py

3 Obsluga bledow

3.1 Kluczowe mechanizmy walidacji

1. **Sprawdzenie podstawowych warunkow:**
   - Dane wejsciowe nie moga byc puste
   - Liczba n musi byc calkowita i ≥ 1
   - Musza byc podane wszystkie linie: n, x, y
2. **Walidacja wezlow x:**
   - Wartosci musza byc liczbowe
   - Liczba wezlow musi wynosic dokladnie $n + 1$
   - Wezly musza byc unikalne (brak powtorzen)
3. **Walidacja wartosci y:**
   - Wartosci musza byc liczbowe
   - Liczba wartosci musi wynosic dokladnie $n + 1$
4. **Walidacja punktow t:**
   - Wartosci musza byc liczbowe
   - Musi istniec co najmniej jeden punkt t

3.2 Przyklady bledow i komunikaty

```
Blad: Brak danych wejsciowych.
Blad: Pierwsza linia musi zawierac liczbe calkowita n.
Blad: n musi byc >= 1.
Blad: Zbyt malo linii danych - oczekiwane: n, x, y.
Blad: Linia 2 musi zawierac wartosci liczbowe x.
Blad: Dla n=2 nalezy podac dokladnie 3 wartosci x (podano 2).
Blad: Wartosci x musza byc unikalne.
Blad: Linia 3 musi zawierac wartosci liczbowe y.
Blad: Dla n=2 nalezy podac dokladnie 3 wartosci y (podano 4).
Blad: W liniach z t musza znajdowac sie liczby.
Blad: Brak punktow t do obliczenia interpolacji.
```

4 Przyklad dzialania

4.1 Dane wejsciowe (poprawne)

```
2
0.0 1.0 2.0
1.0 2.0 5.0
0.5 1.5
```

4.2 Wynik

```
1.25
3.25
```

4.3 Bledne dane wejsciowe i komunikaty

```
2
0.0 1.0 0.0
1.0 2.0 5.0
0.5 1.5

Blad: Wartosci x musza byc unikalne.
```

```
2
0.0 1.0 2.0
1.0 2.0
0.5 1.5

Blad: Dla n=2 nalezy podac dokladnie 3 wartosci y (podano 2).
```

```
2
0.0 1.0 2.0
1.0 2.0 5.0

Blad: Brak punktow t do obliczenia interpolacji.
```

5 Podsumowanie

- Program **sprawdza poprawnosc danych** przed przystapieniem do obliczen
- **Precyzyjne komunikaty bledow** wskazuja, co jest nie tak
- **Oddzielny modul walidacji** zapewnia czytelnosc kodu
- Program **nie przerywa dzialania** w przypadku bledow - wyswietla komunikat i konczy prace