Data wykonania: 13.03.2024

INTERPOLACJA NEWTONA

Metody numeryczne

1. Wprowadzenie teoretyczne

Wzór interpolacyjny Lagrange'a może sprawiać pewne trudności w praktyce, szczególnie gdy zmienia się liczba węzłów. Aby obliczyć wielomian o określonym stopniu, musimy zaczynać od początku. Dodawanie nowych węzłów interpolacyjnych nie pozwala na modyfikację wcześniej obliczonego wielomianu Lagrange'a. Na szczęście wzór interpolacyjny Newtona, który jest równoważny wzorowi Lagrange'a, eliminuje tę niedogodność.

Niech (x_0, x_1, …, x_n) będą punktami interpolacji, w których wielomian interpolacyjny przyjmuje wartości (y_0, y_1, …, y_n). Wówczas możemy wprowadzić pojęcie **ilorazów różnicowych**.

- ***Pierwszego rzędu***

$$[x_0, x_1] = \frac{y_1 - y_0}{x_1 - x_0} \qquad [x_1, x_2] = \frac{y_2 - y_1}{x_2 - x_1} \qquad [x_{n-1}, x_n] = \frac{y_n - y_{n-1}}{x_n - x_{n-1}}$$

- ***Drugiego rzędu***

$$[x_0, x_1, x_2] = \frac{[x_1 - x_2] - [x_0 - x_1]}{x_2 - x_0}$$

$$[x_1, x_2, x_3] = \frac{[x_2 - x_3] - [x_1 - x_2]}{x_3 - x_1}$$

$$[x_{n-2}, x_{n-1}, x_n] = \frac{[x_{n-1} - x_n] - [x_{n-2} - x_{n-1}]}{x_n - x_{n-2}}$$

- ***k-rzędu***

$$[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{[x_{i+1}, x_{i+2}, \ldots, x_{i+k}] - [x_i, x_{i+1}, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$$ dla k = 1,2,...oraz i – 0,1,2,…

Wzór interpolacyjny Newtona

$$W_n(x) = y_0 + [x_0, x_1](x - x_0) + [x_0, x_1, x_2](x - x_0)(x - x_1) + [x_0, x_1, \ldots, x_n](x - x_0)(x - x_1) \cdot \ldots \cdot (x - x_{n-1})$$

n → ilość rzędów

x → punkt dla którego otrzymamy przybliżoną wartość funkcji

2. Opis implementacji numerycznej

Danymi wejściowymi do tej metody są węzły interpolacji (x_1, x_2, … , x_n), w których wielomian interpolacyjny przyjmuje odpowiednio wartości y_0, y_1, … , y_n.

```
cout << "\nWartosci x oraz y:" << endl;
for (int i = 0; i < ilosc_wartosci_xy; i++) {
    tabx[i] = stod(tab[2 * i]);
    cout << "x" << i + 1 << " = " << tabx[i];
    taby[i] = stod(tab[2 * i + 1]);
    cout << "; y" << i + 1 << " = " << taby[i];
    cout << endl;
}
```

Rysunek 1: Fragment kodu (przypisywanie danych)

```
Wartosci x oraz y:
x1 = -2; y1 = -1
x2 = -1; y2 = 0
x3 = 0; y3 = 5
x4 = 2; y4 = 99
x5 = 4; y5 = -55
```

Rysunek 1.1: Fragment konsoli (wyświetlenie danych odczytanych z pliku)

Dodatkowo aby usprawnić testowanie programu, dane te wczytywane są z pliku dzięki użyciu biblioteki **fstream**.

```
ifstream plik("metoda2.txt");
string* tab = new string[liczbaLinii];
for (int i = 0; i < liczbaLinii; i++) {
    getline(plik, tab[i]);
}
plik.close();
```

Rysunek 2: Fragment kodu (odczytanie poszczególnych linii z pliku do elementów tablicy tab)

```
// Obliczanie wartości interpolowanej
for (int i = 0; i < ilosc_wartosci_xy - 1; i++) {
    tab1[i] = (taby[i + 1] - taby[i]) / (tabx[i + 1] - tabx[i]);
}
wynik += (tab1[0] * (punkt_x - tabx[0]));

for (int i = 0; i < ilosc_wartosci_xy - 2; i++) {
    double temp = 1;
    for (int j = 0; j < (ilosc_wartosci_xy - (i + 2)); j++) {
        tab1[j] = (tab1[j + 1] - tab1[j]) / (tabx[j + i + 2] - tabx[j]);
    }
    for (int k = 0; k < i + 2; k++) {
        temp *= (punkt_x - tabx[k]);
    }
    wynik += (tab1[0] * temp);
}
```

Rysunek 3: Fragment kodu (obliczanie wartości interpolowanej)

1. Tworzymy tablicę ***tab1***, w której przechowujemy kolejne wyniki ilorazów różnicowych.

2. W pierwszej pętli for obliczamy ilorazy różnicowe pierwszego rzędu na podstawie podanych wartości ***tabx*** oraz ***taby***.

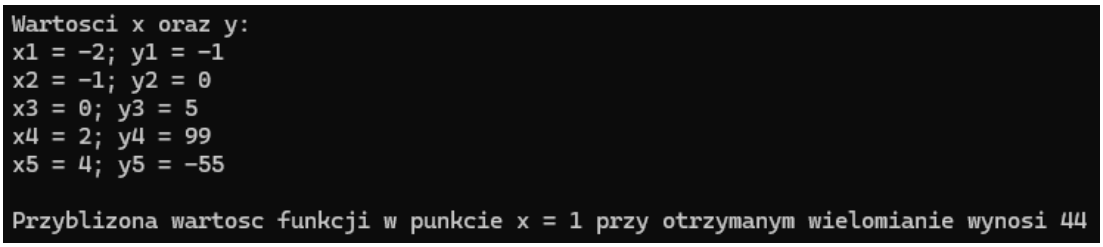
3. Następnie korzystając z tych danych, w drugiej pętli for obliczamy wartość interpolowaną dla danego punktu ***punkt_x***, wykorzystując wzory k-rzędu ilorazów różnicowych.

4. Przy końcu każdej iteracji drugiej pętli for zliczany jest wynik do zmiennej ***wynik*** ze wzoru interpolacyjnego Newtona.

3. Testy kodu numerycznego

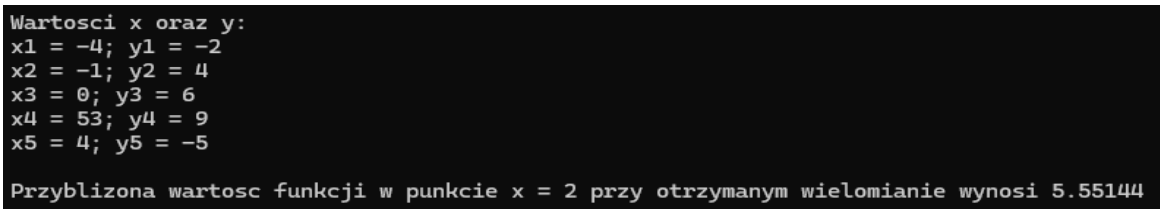
Testy zostały przeprowadzone, aby sprawdzić działanie programu.

Test I: dla zbioru punktów {(-2, -1), (-1, 0), (0, 5), (2, 99), (4, -55)} i X = 1



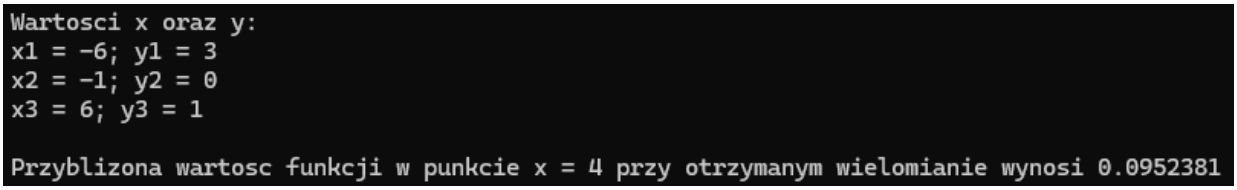
Rysunek 4: test *dla x = 1*

Test II: dla zbioru punktów {(-4, -2), (-1, 4), (0, 6), (53, 9), (4, -5)} i X = 2



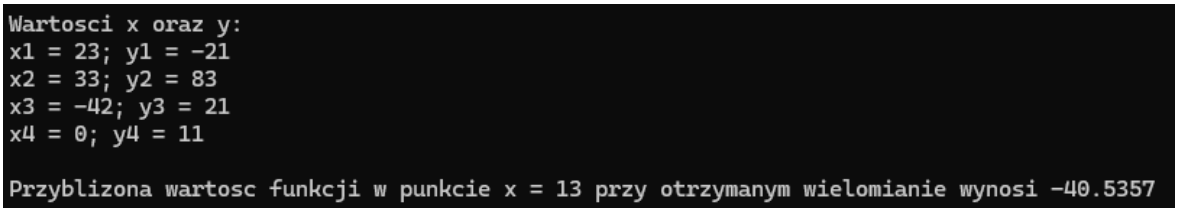
Rysunek 5: test dla x = 2

Test III: dla zbioru punktów {(-6, 3, (-1, 0), (6, 1)} i X = 4



Rysunek *6: test dla x = 4*

Test IV: dla zbioru punktów {(23, -21), (33, 83), (-42, 21), (0, 11)} i X = 13



Rysunek 7: test dla x = 13

Zostały przeprowadzone 4 testy jednostkowe dla zbiorów liczb zawierających 3, 4, 5 elementów. Przy każdym teście program podawał poprawne wyniki.

4. Wnioski

Interpolacja Newtona to skuteczna metoda numeryczna do przybliżania funkcji na podstawie punktów danych. W niniejszym opracowaniu omówiono teorię interpolacji Newtona i przedstawiono implementację tej metody. Przeprowadzone testy jednostkowe potwierdziły poprawność działania zaimplementowanej metody dla różnych zestawów danych wejściowych oraz wartości X. Wyniki testów były zgodne z oczekiwanymi rezultatami, co świadczy o prawidłowym działaniu programu w różnych warunkach. Wyniki testów oraz funkcjonalność programu potwierdzają skuteczność interpolacji Newtona w przybliżaniu funkcji na podstawie ograniczonej liczby punktów danych