Sprawozdanie z laboratorium nr 3

Interpolacja Newtona

Wojciech Matys

Wydział WIMiIP

Kierunek: Inżynieria obliczeniowa

Grupa lab nr 4

Data wykonania: 21.03.2024

Cel ćwiczenia

Celem laboratorium było zapoznanie się z metodą interpolacji Newtona oraz implentacja jej w wybranym przez siebie języku, ja zdecydowałem się na C++.

Wstęp teoretyczny

Interpolacja w metodach numerycznych odnosi się do procesu przybliżania nieznanej funkcji za pomocą zestawu znanych danych punktowych. W skrócie, interpolacja polega na konstrukcji funkcji, która "przechodzi przez" dane punkty, pozwalając na szacowanie wartości funkcji w punktach pomiędzy tymi znanymi. Jest to jedna z podstawowych technik analizy danych numerycznych i jest szeroko stosowana w różnych dziedzinach nauki, inżynierii i informatyki.

Interpolacja w metodach numerycznych jest kluczowym narzędziem w analizie danych, wizualizacji, modelowaniu matematycznym oraz w praktycznych zastosowaniach, takich jak przetwarzanie obrazów, grafika komputerowa czy symulacje fizyczne. Jednak zawsze warto pamiętać, że dobór odpowiedniej metody interpolacji oraz odpowiednia analiza uzyskanych wyników są kluczowe dla uzyskania wiarygodnych i użytecznych wyników. Istnieje wiele różnych metod interpolacji, w tym sprawozdaniu przeanalizuję metodę interpolacji Newtona.

Interpolacja Newtona

Interpolacja Newtona jest metodą numeryczną używaną do przybliżania funkcji za pomocą wielomianu interpolacyjnego, który przechodzi przez zestaw punktów danych. Ta metoda opiera się na interpolacji wielomianowej za pomocą tzw. różnic dzielonych Newtona.

Jak działa interpolacja Newtona:

1. **Określenie punktów danych:** Na początku, dane są zbierane w postaci par (x, y), gdzie x to argumenty, a y to odpowiadające im wartości funkcji.
2. **Obliczenie różnic dzielonych** Newtona: Różnice dzielone Newtona to współczynniki wielomianu interpolacyjnego. Wartości te są obliczane rekurencyjnie na podstawie punktów danych.
   Definiujemy je jako:

   - Pierwsza różnica dzielona:
     $f[x_i] = y_i$
   - Kolejne różnice dzielone:
     $f[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, \ldots, x_{i+k}] - f[x_i, x_{i+1}, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$

   Gdzie:

   - $x_i, x_{i+1}, \ldots, x_{i+k}$ to różne wartości argumentów.
   - $f[x_i, x_{i+1}, \ldots, x_{i+k}]$ to różnica dzielona Newtona dla tych wartości argumentów.

3. **Konstrukcja wielomianu interpolacyjnego**: Po obliczeniu różnic dzielonych Newtona, możemy skonstruować wielomian interpolacyjny za pomocą wzoru:

$$P(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \ldots$$

Gdzie:

- $P(x)$ to wielomian interpolacyjny Newtona.
- $x_0, x_1, \ldots$ to argumenty danych punktów.
- $f[x_i, x_{i+1}, \ldots, x_{i+k}]$ to różnice dzielone Newtona obliczone dla odpowiednich argumentów.

4. **Obliczenie wartości wielomianu**: Wreszcie, aby uzyskać przybliżoną wartość funkcji w punkcie x, wystarczy podstawić x do wielomianu $P(x)$ i obliczyć wartość.

Wzór ten pozwala na wyznaczenie wielomianu interpolacyjnego Newtona, który jest stosunkowo łatwy do obliczenia i implementacji numerycznej.

Jeśli n oznacza liczbę punktów danych, to złożoność obliczeniowa interpolacji Newtona wynosi $O(n^2)$, co oznacza, że może być bardziej wydajny niż metody o podobnej dokładności, takie jak interpolacja Lagrange'a, zwłaszcza dla większych zbiorów danych.

Implementacja

Interpolacje Newtona implementuję w języku programowania c++, korzystając z środowiska Visual Studio

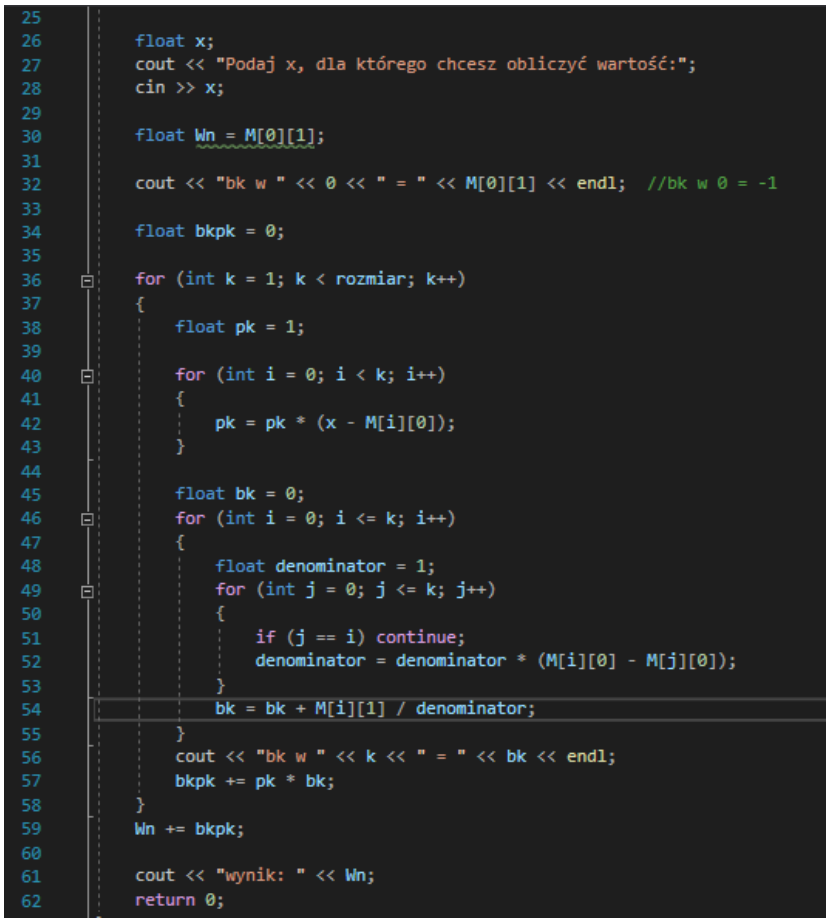
[1] Pierwsza część programu

```cpp
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    int rozmiar;
    fstream czytaj;
    czytaj.open("MN2.txt");
    czytaj >> rozmiar;
    cout << rozmiar;

    int** M = new int* [rozmiar];
    for (int i = 0; i < rozmiar; i++)
        M[i] = new int[rozmiar];
    cout << endl;
    for (int i = 0; i < rozmiar; i++)
    {
        for (int j = 0; j < 2; j++)
        {
            czytaj >> M[i][j];
            cout << M[i][j] << " ";
        }
        cout << endl;
    }
```

[1] Ta część kodu odpowiada za odczytanie danych z pliku tekstowego i zapisanie ich w tablicy dwuwymiarowej. Rozmiar tablicy jest odczytywany z pierwszej linii pliku, a pozostałe dane są odczytywane wiersz po wierszu. Odczytane dane są wyświetlane na ekranie.

[2] Druga część programu, (implementacja algorytmu)



```
25
26      float x;
27      cout << "Podaj x, dla którego chcesz obliczyć wartość:";
28      cin >> x;
29
30      float Wn = M[0][1];
31
32      cout << "bk w " << 0 << " = " << M[0][1] << endl;  //bk w 0 = -1
33
34      float bkpk = 0;
35
36      for (int k = 1; k < rozmiar; k++)
37      {
38          float pk = 1;
39
40          for (int i = 0; i < k; i++)
41          {
42              pk = pk * (x - M[i][0]);
43          }
44
45          float bk = 0;
46          for (int i = 0; i <= k; i++)
47          {
48              float denominator = 1;
49              for (int j = 0; j <= k; j++)
50              {
51                  if (j == i) continue;
52                  denominator = denominator * (M[i][0] - M[j][0]);
53              }
54              bk = bk + M[i][1] / denominator;
55          }
56          cout << "bk w " << k << " = " << bk << endl;
57          bkpk += pk * bk;
58      }
59      Wn += bkpk;
60
61      cout << "wynik: " << Wn;
62      return 0;
```

[2]. Ten algorytm implementuje interpolację Newtona w celu obliczenia wartości funkcji dla danego x na podstawie podanych punktów (x, y) zawartych w tablicy dwuwymiarowej `M`.

Działanie algorytmu można opisać w następujących punktach:

1. Użytkownik zostaje poproszony o podanie wartości `x`, dla której chce obliczyć wartość funkcji.

2. Inicjalizowana jest zmienna `Wn` jako pierwszy element `y` z tablicy `M`.

3. Wyświetlana jest informacja o wartości `bk` dla `k = 0`, która jest równa pierwszej wartości `y` z tablicy `M`.

4. Zmienna `bkpk` jest inicjalizowana na wartość 0.

5. Pętla `for` iteruje po wszystkich pozostałych elementach tablicy `M`.

- Dla każdej iteracji pętli obliczana jest wartość `pk`, która jest iloczynem różnic `(x - xi)` dla `i` od 0 do `k - 1`.

- Następnie obliczana jest wartość `bk` za pomocą kolejnej pętli `for`.

- Dla każdej iteracji tej pętli obliczany jest mianownik `denominator` za pomocą iloczynu różnic `(xi - xj)` dla `j` od 0 do `k`.

- Wartość `bk` jest zwiększana o iloraz `yi / denominator`, gdzie `yi` to wartość `y` dla `i`-tego punktu.

- Wyświetlana jest informacja o wartości `bk` dla obecnej iteracji `k`.

- Obliczana jest wartość `bkpk` poprzez dodanie iloczynu `pk * bk` do dotychczasowej sumy.

6. Po zakończeniu pętli, wartość `Wn` jest aktualizowana przez dodanie wartości `bkpk`.

7. Wyświetlany jest wynik obliczeń jako `Wn`, co odpowiada wartości funkcji dla podanego `x`.

Legenda do frgamentu drugiego kodu [2]

1. `x`: Reprezentuje wartość, dla której użytkownik chce obliczyć wartość funkcji.

2. `Wn`: Reprezentuje wartość wielomianu interpolacyjnego Newtona. Jest to suma dotychczasowych obliczeń.

3. `M`: Jest to tablica dwuwymiarowa zawierająca punkty (x, y), na podstawie których będzie wykonywana interpolacja.

4. `rozmiar`: Reprezentuje rozmiar tablicy dwuwymiarowej `M`.

5. `bk`: Reprezentuje kolejne współczynniki interpolacyjne wyznaczane w każdej iteracji.

6. `pk`: Reprezentuje część wielomianu zależną od `x` w danym kroku iteracji.

7. `bkpk`: Jest to suma iloczynów `pk * bk` obliczanych w każdej iteracji pętli dla aktualnej wartości `x`.

8. `i`, `j`, `k`: Są to zmienne pomocnicze używane do iteracji po elementach tablicy `M` oraz do wyznaczania współczynników i części wielomianu interpolacyjnego.

Testy jednostkowe

Test 1 (Przykład z instrukcji, sprawdzamy czy nasz program działa poprawnie)

Chcemy wyznaczyć wartość funkcji w punkcie **x = 1**

Dane punkty: (-2, -1); (-1, 0); (0, 5); (2, 99); (4, -55)

Dane wejściowe z pliku, który odczytuje program:

5

-2 -1

-1 0

0 5

2 99

4 -55

Wynik programu:

```
5
-2 -1
-1 0
0 5
2 99
4 -55
Podaj x, dla ktorego chcesz obliczyc wartosc:1
bk w 0 = -1
bk w 1 = 1
bk w 2 = 2
bk w 3 = 3
bk w 4 = -2
wynik: 44
```

Wynik programu = 44

Oczekiwany wynik = 44

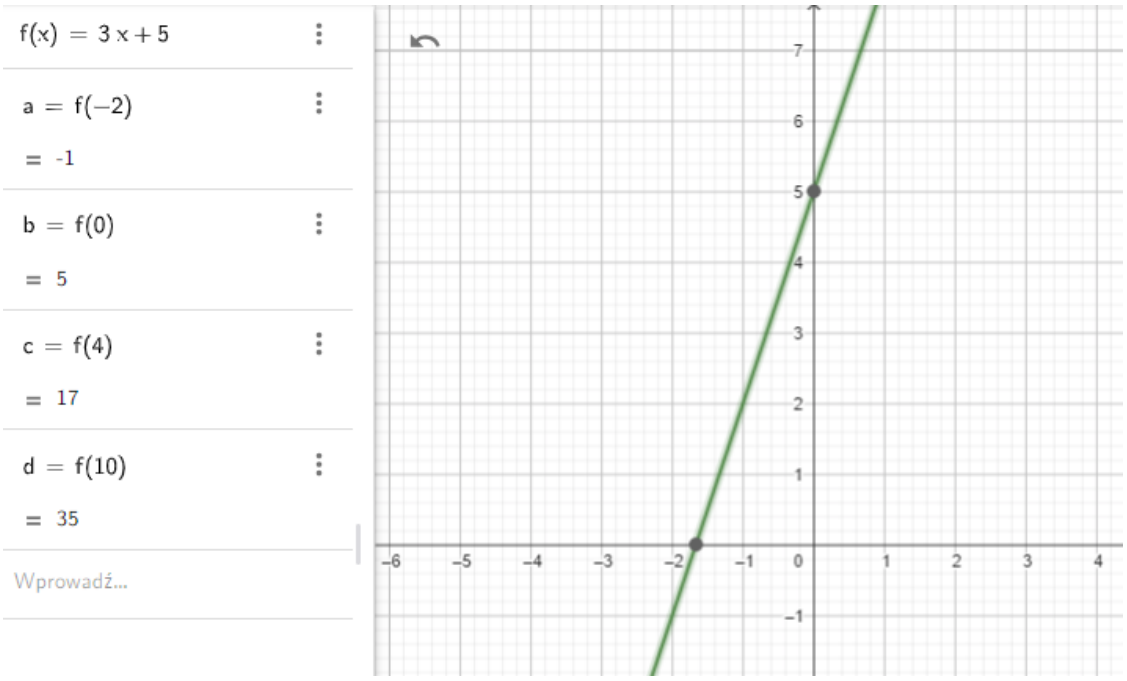
Wynik programu = Oczekiwany wynik

Na potrzeby, sprawdzenia poprawności działania algorytmu wypisałem w konsoli, również współczynniki interpolacyjne. Zgadzają się z podanymi w prezentacji.

Test 2

Chcemy wyznaczyć wartość funkcji w punkcie **x = 7**

Nasz zestaw danych:



Dane wejściowe z pliku, który odczytuje program:

4

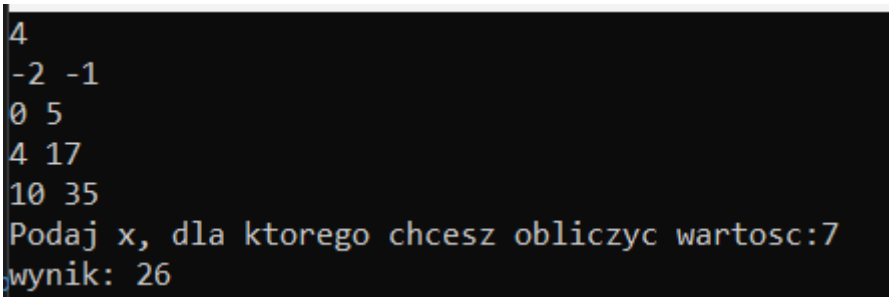
-2 -1

-0 5

4 17

10 35

Wynik programu:



Wynik programu = 26

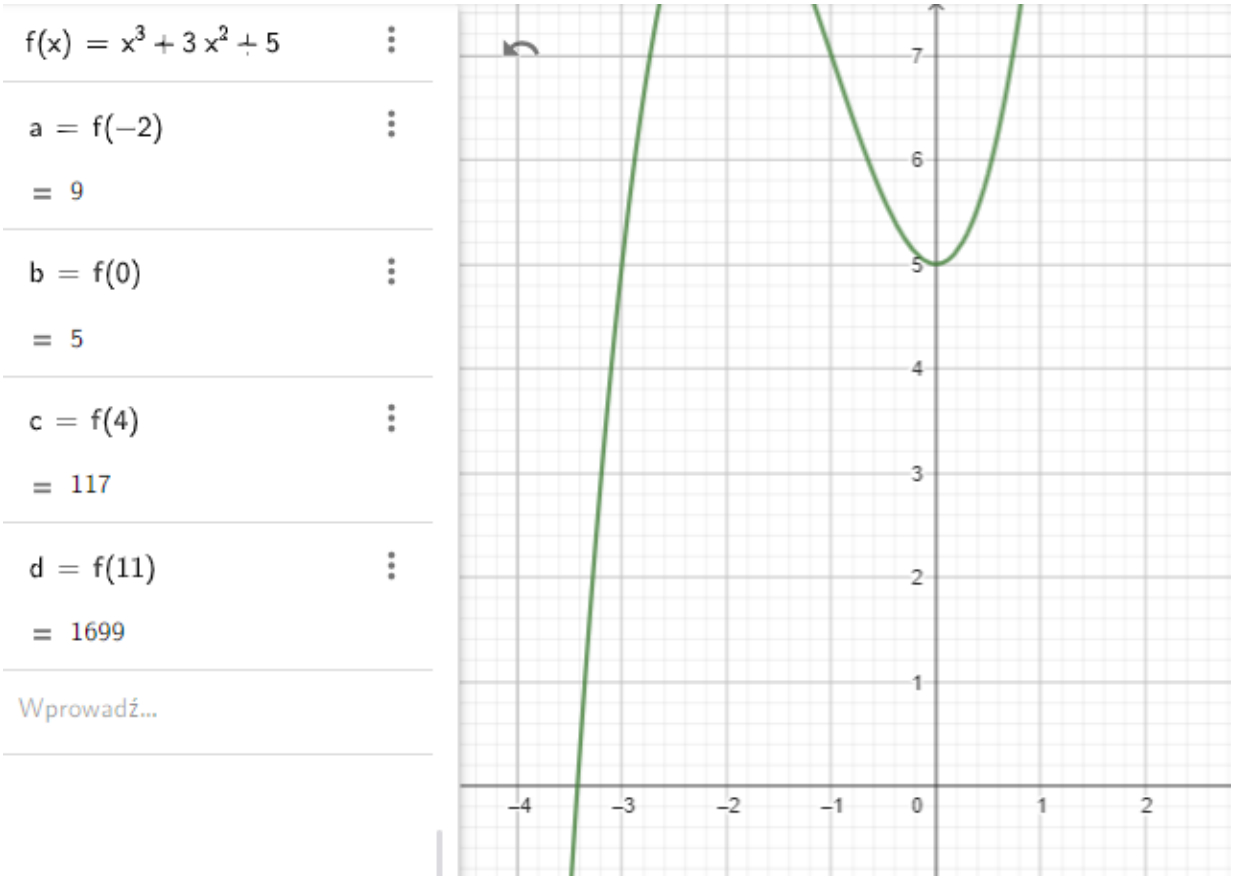
Oczekiwany wynik = 26

Wynik programu = Oczekiwany wynik

Test 3

Chcemy wyznaczyć wartość funkcji w punkcie **x = 9**

Nasz zestaw danych:



Dane wejściowe z pliku, który odczytuje program:

4

-2 9

0 5

4 117

11 1699

Wynik programu:

```
4
-2 9
0 5
4 117
11 1699
Podaj x, dla ktorego chcesz obliczyc wartosc:9
wynik: 977
```

Wynik programu = 977

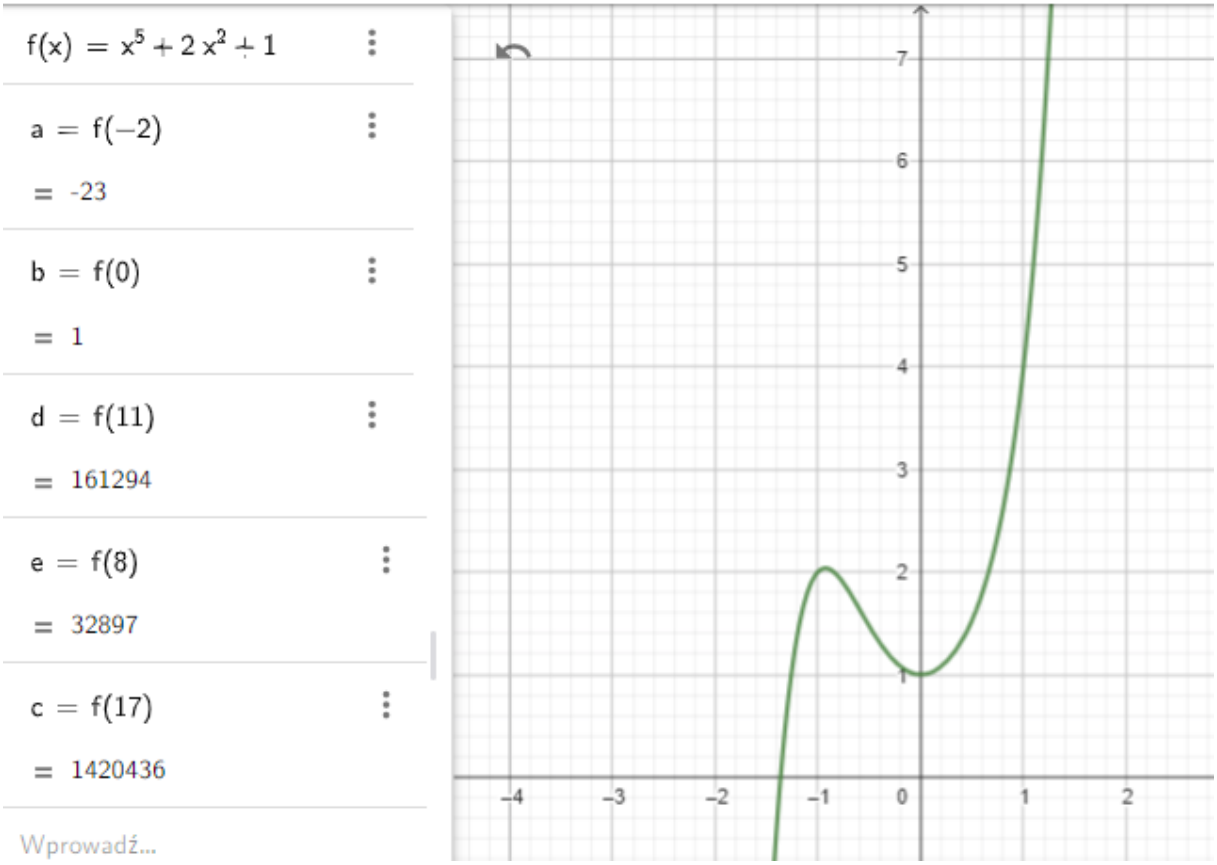
Oczekiwany wynik = 977

Wynik programu = Oczekiwany wynik

Test 4

Chcemy wyznaczyć wartość funkcji w punkcie x = 15

Nasz zestaw danych:



Dane wejściowe z pliku, który odczytuje program:

5

-2 -23

0 1

11 161294

8 32897

17 1420436

Wynik programu:

```
5
-2 -23
0 1
11 161294
8 32897
17 1420436
Podaj x, dla ktorego chcesz obliczyc wartosc:15
wynik: 774106
```

Wynik programu = 774106

Oczekiwany wynik = 759826

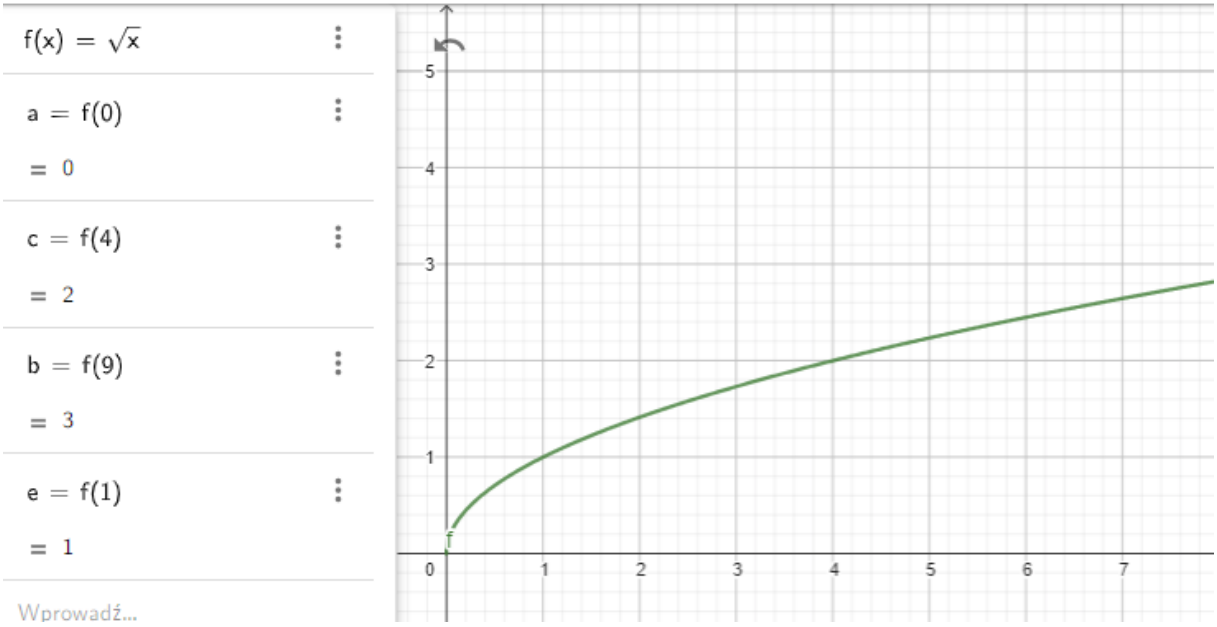
Wynik programu != Oczekiwany wynik

Skala błędu wynosi około 1,8 %

Test 5

Chcemy wyznaczyć wartość funkcji w punkcie x = 8

Nasz zestaw danych:



Dane wejściowe z pliku, który odczytuje program:

4

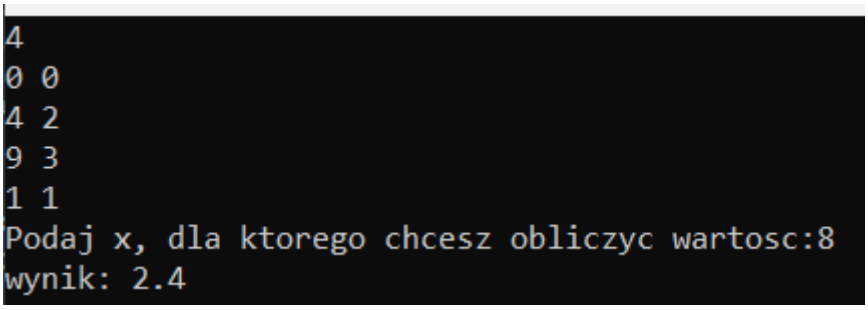
0 0

4 2

9 3

1 1

Wynik programu:



Wynik programu = 2.4

Oczekiwany wynik = 2.828

Wynik programu != Oczekiwany wynik

Skala błędu wynosi około 15 %

Wnioski

Interpolacja Newtona to wszechstronna metoda używana do przybliżania wartości funkcji na podstawie danych punktów. Oto wnioski dotyczące jej stosowania:

1. Sprawdza się dobrze dla funkcji gładkich: Interpolacja Newtona jest skuteczna dla funkcji, które są gładkie i dobrze zachowują się w otoczeniu punktów danych. Oznacza to, że dla funkcji, których pochodne są ciągłe i dobrze zachowują się w obrębie przedziału, interpolacja Newtona może dostarczyć dokładne przybliżenie.

2. Może dawać złe wyniki dla funkcji o dużej zmienności: Dla funkcji, które mają dużą zmienność lub nie są gładkie, interpolacja Newtona może dawać gorsze wyniki, zwłaszcza gdy punkty danych są rzadko rozmieszczone lub występują duże oscylacje w wartościach funkcji między punktami danych.

3. Wartość liczby węzłów: Wielkość zbioru danych, czyli liczba punktów, ma istotne znaczenie dla dokładności interpolacji Newtona. Zwiększenie liczby węzłów (punktów danych) zazwyczaj prowadzi do dokładniejszego przybliżenia funkcji. Jednakże zbyt duża liczba węzłów może prowadzić do zjawiska znanego jako efekt Rungego, gdzie interpolacja wielomianowa może pogorszyć się na końcach przedziału.

4. Ważne jest równomiernie rozmieszczenie punktów: Aby uzyskać najlepsze wyniki interpolacji Newtona, ważne jest równomierne rozmieszczenie punktów danych w obszarze, w którym funkcja jest interpolowana. W przeciwnym razie może to prowadzić do nadmiernego dopasowania się do konkretnych wartości punktów danych i nieprawidłowości w interpolacji.

5. Zastosowanie w praktyce: Interpolacja Newtona jest powszechnie stosowana w praktyce, szczególnie w dziedzinach takich jak nauki przyrodnicze, inżynieria, ekonomia i informatyka. Jest używana do analizy danych eksperymentalnych, interpolacji krzywych i przybliżania wartości funkcji w różnych aplikacjach.

Podsumowując, interpolacja Newtona jest użyteczną metodą do przybliżania funkcji, zwłaszcza gdy funkcje są gładkie i gdy punkty danych są równomiernie rozmieszczone. Jednakże należy ostrożnie dobierać liczbę węzłów, aby uniknąć nadmiernego dopasowania się do danych lub efektu Rungego.

Źrodła

-wykresy funkcji pobrane ze strony https://www.geogebra.org/?lang=pl