Imię i nazwisko Meg Paskowski	**Temat laboratorium** Interpolacja Lagrange'a	**Data oddania** 18.03.2024r.	**Data ćwiczeń** 08.03.2024r.
Prowadzący dr hab. inż. Marcin Hojny		**Grupa laboratoryjna** 4	

1. Cel ćwiczenia

Celem laboratorium nr. 1 było zapoznanie z Interpolacją Lagrange'a oraz implementacja zagadnienia w wybranym przez siebie języku programowania.

2. Wstęp teoretyczny

Interpolacja to metoda numeryczna polegająca na przybliżaniu funkcji za pomocą wielomianu Lagrange'a stopnia n, który przyjmuje te same wartości co funkcja pierwotna w n+1 punktach (węzłach interpolacji). Według twierdzenia Weierstrassa, dla każdej funkcji y=f(x), która jest ciągła na przedziale zamkniętym [a,b], istnieje wielomian o dostatecznie wysokim stopniu, który może dokładnie przybliżyć tę funkcję.

Kluczowym elementem tej metody jest wielomian Lagrange'a, który tworzymy tak, aby precyzyjnie dopasować się do naszych danych. Wielomian ten jest wyrażany za pomocą określonego wzoru:

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

Gdzie:

- **x_i** → jest argumentem dla którego chcemy znaleźć wartość funkcji
- **y_i** → wartość funkcji odpowiadająca argumentowi x_i

Oraz wartość współczynników l_i (wielomianów bazowych), wyznaczamy ze wzoru:

$$l_i(x) = \prod_{0<j<n \,\&\&\, j\#i} \frac{x - x_j}{x_i - x_j}$$

Każdy taki wielomian jest konstruowany w taki sposób, że przyjmuje wartość 1 w punkcie x_i oraz wartość 0 w każdym innym punkcie x_j. W ten sposób, wielomian $l_i(x)$ dokładnie odzwierciedla wartości funkcji w tych konkretnych punktach.

Przykłady zastosowania interpolacji Lagrange'a:

- przybliżanie danych, które nie są wyraźnie opisane, poprzez tworzenie gładkiego wielomianu przechodzącego przez znane punkty danych.
- do przybliżania funkcji, które są trudne do analizy lub obliczenia.
- symulacje, analiza danych, oraz do rozwiązywania równań różniczkowych.
- w modelowaniu danych finansowych, takich jak ceny akcji, stopy procentowe.
- wykorzystuje się do animacji postaci i obiektów w grach komputerowych, aby uzyskać płynne ruchy.

- są wykorzystywane w różnych metodach numerycznych, takich jak: całkowanie numeryczne, różniczkowanie numeryczne.

3. **Implementacja**

W ramach ćwiczeń zaimplementowano metodę interpolacji Lagrange'a w języku C++. Napisany program wczytuje dane z pliku „NM1.txt”.

```
int size;
fstream read("NM1.txt");
```

Następnie sprawdzamy, czy plik został poprawnie otwarty

```
if (read.is_open()) {..}
```

W przeciwnym razie wyświetlony zostanie komunikat o błędnym odczycie pliku.

```
else   cout << "Nie udalo sie otworzyc pliku" << endl;
```

Następnie po pozytywnym otwarciu pliku wczytywana jest liczba punktów funkcji, dla których będziemy wykonywać interpolacje.

```
read >> size;

cout << "Ilosc punktow dla ktorych wykonujemy zadanie: " << size << endl;
```

I tworzona jest tablica dynamiczna „points”, która przechowuje powyższą liczbę punktów, wczytując je po kolei z pliku.

```
//Tworzenie tablicy dynamicznej dwuwymiarowej, przechowującej współrzędne punktów x, y
double** points = new double* [size];

for (int i = 0; i < size; i++)  points[i] = new double[size];

//Pętla dodajaca punkty do tablicy

for (int i = 0; i < size; i++) {
       read >> points[i][0] >> points[i][1];
       cout << "Punkt utworzony M=(" << points[i][0] << ", " << points[i][1] << ")" <<
endl;
}
```

W kolejnym kroku tworze zmienną „value”, która będzie przechowywać wartoś x dla którego szukamy wartości Y funkcji.

```
double value;
cout << "Podaj wartosc X dla ktorego szukamy wartosci Y. " << endl;
cin >> value;
```

Tworze zmienną „result_y”, która będzie służyć do przechowywanie wyniku wyszukanej dla zmiennej Y.

```
double result_y = 0;    //Zmienna przechowująca wynik
```

Następnie przechodzę do głównej funkcji – interpolacji Lagrange'a.
Pierwsza pętla „for” przechodzi przez wszystkie elementy znajdujące się w tablicy punktów „points”. Druga pętla również przechodzi przez każdy punkt, ale wylicza wartość l(value).

```
for (int i = 0; i < size; i++){ // Przechodzimy przez każdy punkt i liczymy wartości
       double wielomian_l = 1;    // Neutralny element mnożenia

      for (int j = 0; j < size; j++){     //Obliczanie l(value).
          if (j != i){
```

```
            // Wartość l obliczona z wzoru opisanego w częsi teorytycznej - na
        wartość współczynników.
            wielomian_l *= (value - points[j][0]) / (points[i][0] - points[j][0]);
          }
      }
      result_y += points[i][1] * wielomian_l;   //Wynikiem jest suma wartości L(value)
= xi * l1(value) + ...
}
```

Na sam koniec należy pamiętać o zwolnieniu pamięci oraz zamknięciu odczytu pliku.

```
for (int i = 0; i < size; ++i)
    delete[] points[i];
delete[] points;

read.close();
```

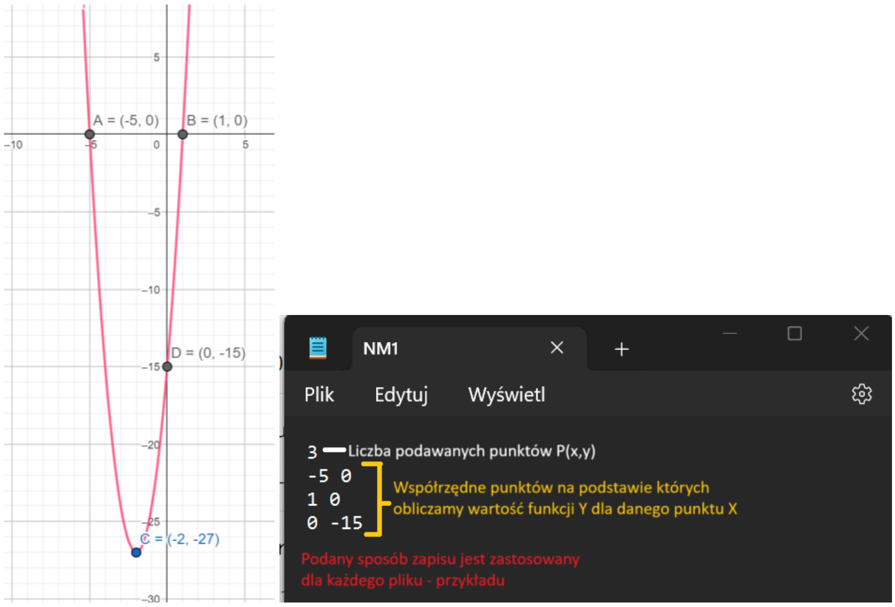
Zdjęcie przedstawia pełną implementacje kodu w języku C++.

```
int size;
fstream read("NM1.txt");

if (read.is_open()) {
    read >> size;   //Liczba punktów
    cout << "Ilosc punktow dla ktorych wykonujemy zadanie: " << size << endl;

    //Tworzenie tablicy dynamicznej dwuwymiarowej, przechowującej współrzędne punktów x, y
    double** points = new double* [size];

    for (int i = 0; i < size; i++)  points[i] = new double[size];

    for (int i = 0; i < size; i++) {
        read >> points[i][0] >> points[i][1];

        cout << "Punkt utworzony M=(" << points[i][0] << ", " << points[i][1] << ")" << endl;
    }

    double value;
    cout << "Podaj wartosc X dla ktorego szukamy wartosci Y. " << endl;
    cin >> value;

    double result_y = 0;   //Zmienna przechowująca wynik

    for (int i = 0; i < size; i++){ // Przechodzimy przez kazdy punkt i liczymy warosci
        double wielomian_l = 1;

        for (int j = 0; j < size; j++){     //obliczanie l(value)
            if (j != i){
                wielomian_l *= (value - points[j][0]) / (points[i][0] - points[j][0]);
            }
        }
        result_y += points[i][1] * wielomian_l;   //Wynikiem jest suma wartości L(value) = xi * l1(value) + ...
    }
    cout << "Dla podanego x= " << value << " wartosc y wynosi: " << result_y << endl;

    for (int i = 0; i < size; ++i)
        delete[] points[i];
    delete[] points;

}else   cout << "Nie udalo sie otworzyc pliku" << endl;

read.close();

system("PAUSE");
return 0;
```

4. Testy

Testy zostały wykonane dla danych z pliku „NM1.txt". Zawartość pliku:

a. Dla funkcji kwadratowej: f(x)=3 x^(2)+12 x-15



Interpretacja graficzna funkcji a oraz zawartość pliku „NM1.txt”

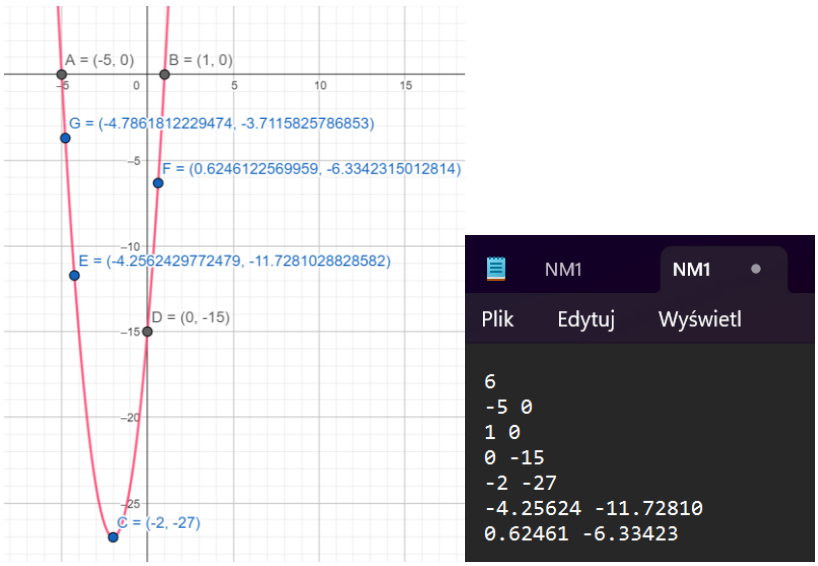
Szukany punkt C dla x = -2

- Oczekiwany wynik: -27
- Otrzymany wynik: -27

```
C:\Users\pasko\source\repos\
Ilosc punktow dla ktorych wykonujemy zadanie: 3
Punkt utworzony M=(-5, 0)
Punkt utworzony M=(1, 0)
Punkt utworzony M=(0, -15)
Podaj wartosc X dla ktorego szukamy wartosci Y.
-2
Dla podanego x= -2 wartosc y wynosi: -27
Press any key to continue . . .
```

Wynik programu po uruchomieniu.

b. Ponownie dla funkcji: f(x)=3 x^(2)+12 x-15



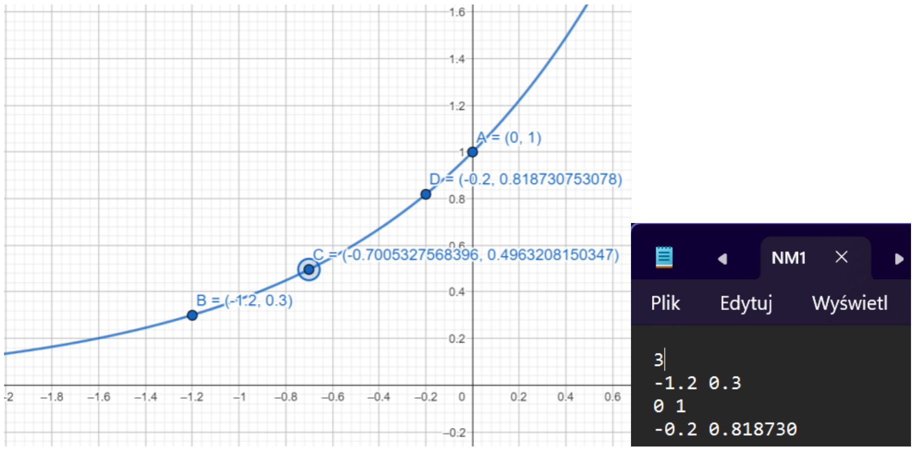
Szukany punkt G dla ok. x = -4.78618 → Sam wynik przybliżam do 4 liczby po przecinku.

- Oczekiwany wynik: -3.7116
- Otrzymany wynik: -3.7116

```
C:\Users\pasko\source\repos\
Ilosc punktow dla ktorych wykonujemy zadanie: 6
Punkt utworzony M=(-5, 0)
Punkt utworzony M=(1, 0)
Punkt utworzony M=(0, -15)
Punkt utworzony M=(-2, -27)
Punkt utworzony M=(-4.25624, -11.7281)
Punkt utworzony M=(0.62461, -6.33423)
Podaj wartosc X dla ktorego szukamy wartosci Y.
-4.78618
Dla podanego x= -4.78618 wartosc y wynosi: -3.7116
Press any key to continue . . .
```

Wynik programu po uruchomieniu.

c. Dla funkcji eksponencjalnej: $f(x)= e^x$



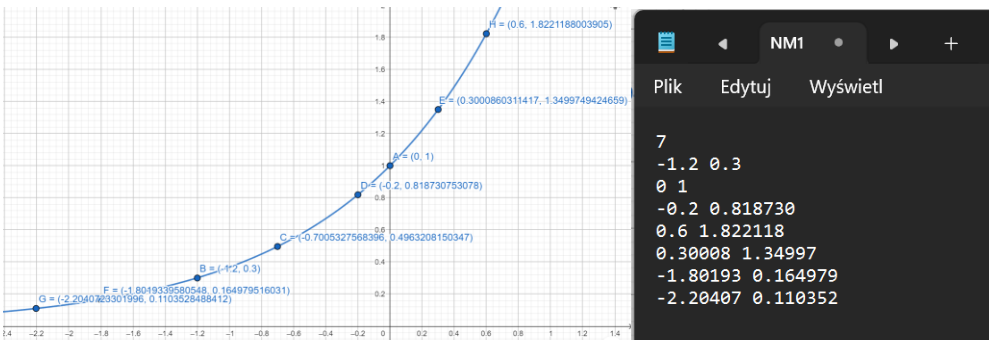
Szukany punkt C dla ok. x = -0.70053

- Oczekiwany wynik: ok. 0.4963
- Otrzymany wynik: ok. 0.478336 → występuje niewielka różnica wyników

```
Ilosc punktow dla ktorych wykonujemy zadanie:
 3
Punkt utworzony M=(-1.2, 0.3)
Punkt utworzony M=(0, 1)
Punkt utworzony M=(-0.2, 0.81873)
Podaj wartosc X dla ktorego szukamy wartosci
Y.
-0.70053
Dla podanego x= -0.70053 wartosc y wynosi: 0.
478336
Press any key to continue . . .
```

Wynik programu po uruchomieniu.

d. Dla funkcji eksponencjalnej: $f(x)= e^x$



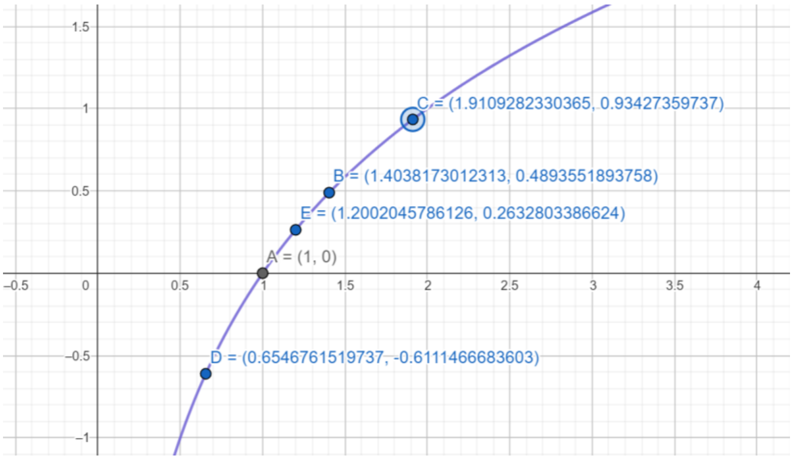
Szukany punkt C dla ok. x = -0.70053

- Oczekiwany wynik: ok. 0.4963
- Otrzymany wynik: ok. 0.495822 → uzyskany wynik przy większej licznie podanych punktów jest jeszcze bardziej zbliżony do oczekiwanego.

```
C:\Users\pasko\source\repo
Ilosc punktow dla ktorych wykonujemy zadanie: 7
Punkt utworzony M=(-1.2, 0.3)
Punkt utworzony M=(0, 1)
Punkt utworzony M=(-0.2, 0.81873)
Punkt utworzony M=(0.6, 1.82212)
Punkt utworzony M=(0.30008, 1.34997)
Punkt utworzony M=(-1.80193, 0.164979)
Punkt utworzony M=(-2.20407, 0.110352)
Podaj wartosc X dla ktorego szukamy wartosci Y.
-0.70053
Dla podanego x= -0.70053 wartosc y wynosi: 0.495
822
Press any key to continue . . .
```

Wynik programu po uruchomieniu.

e. Dla funkcji logarytmicznej: $f(x)=\log_2 x$




```
NM1
Plik   Edytuj   Wyświetl

4
1 0
0.654676 -0.611146
1.20020 0.26328
1.403817 0.489355
```

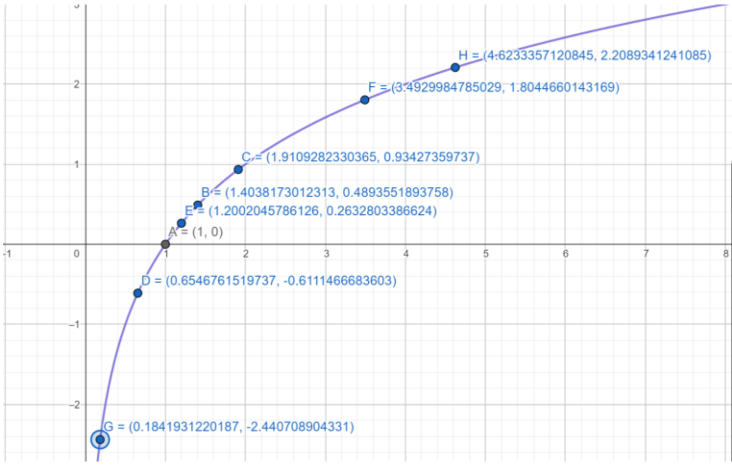
Szukany punkt C dla ok. x = 1.910928

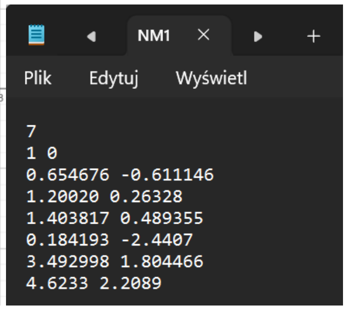
- Oczekiwany wynik: ok. 0.93427
- Otrzymany wynik: ok. 1.01265 → wynik podobny

```
C:\Users\pasko\source\repos\
Ilosc punktow dla ktorych wykonujemy zadanie: 4
Punkt utworzony M=(1, 0)
Punkt utworzony M=(0.654676, -0.611146)
Punkt utworzony M=(1.2002, 0.26328)
Punkt utworzony M=(1.40382, 0.489355)
Podaj wartosc X dla ktorego szukamy wartosci Y.
1.910928
Dla podanego x= 1.91093 wartosc y wynosi: 1.01265
Press any key to continue . . .
```

Wynik programu po uruchomieniu.

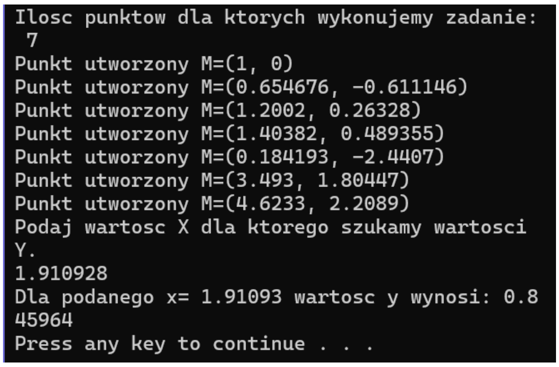
f. Dla funkcji logarytmicznej: $f(x)=\log_2 x$





Szukany punkt C dla ok. x = 1.910928

- Oczekiwany wynik: ok. 0.93427
- Otrzymany wynik: ok. 0.845964 → wynik jest zbliżony



Wynik programu po uruchomieniu.

Jak można zauważyć w przeprowadzonych powyżej testach, wpływ na dokładność wyniku oczekiwanego względem otrzymanego ma ilość punktów oraz ich gęstość ułożenia na danym odcinku.

5. Wnioski

Laboratorium umożliwiło mi zgłębienie wiedzy na temat interpolacji Lagrange'a, metody numerycznej służącej do przybliżania funkcji za pomocą wielomianu stopnia n. Praktyczne zaimplementowanie interpolacji w języku programowania C++, pozwoliło mi lepiej zrozumieć działanie tej metody oraz jak wykorzystać ją w praktyce. Natomiast wykonanie testów jednostkowych dla zaimplementowanego programu pozwoliło na upewnienie się, że działanie kodu jest poprawne i zgodne z oczekiwaniami.

6. Źródła

- Prezentacja autorstwa dr hab. inż. Marcina Hojnego „Metody numeryczne Interpolacja Lagrange'a".