Imię i nazwisko Meg Paskowski	Temat laboratorium Aproksymacja – metoda najmniejszych kwadratów	Data oddania 02.04.2024r.	Data ćwiczeń 22.03.2024r.
Prowadzący dr hab. inż. Marcin Hojny		Grupa laboratoryjna 4	

1. Cel ćwiczenia

Celem laboratorium nr. 4 było zapoznanie się z pojęciem aproksymacji oraz implementacja tego zagadnienia metodą najmniejszych kwadratów w wybranym przez siebie języku programowania.

2. Wstęp teoretyczny

Aproksymacja to technika matematyczna, która polega na przybliżaniu złożonych lub nieregularnych zbiorów danych za pomocą funkcji lub modeli. Pozwala na znalezienie odpowiedniej funkcji lub modelu, który najlepiej pasuje do zestawu danych – zbioru punktów doświadczalnych, co ułatwia analizę, prognozowanie i zrozumienie związków między danymi. Aproksymacja wykorzystywana jest w wielu dziedzinach takich jak nauki ścisłe, inżynieria, ekonomia czy analiza danych.

Podczas zajęć laboratoryjnych zapoznaliśmy się z metodą małych kwadratów. Metoda ta ma na celu znalezienie funkcji, optymalnego dopasowania prostej linii, który najlepiej odzwierciedla podany zestaw danych. Wyznaczona prosta jest opisana wzorem:

$$y = ax + b \tag{1}$$

Gdzie:

- x → zmienna zależna
- y → zmienna niezależna
- a → współczynnik kierunkowy prostej
- b → wyraz wolny

Poszukiwanie parametrów *a* oraz *b* do prostej, tak aby przechodziła ona możliwie jak najbliżej wszystkich punktów doświadczalnych jest realizowana za pomocą minimalizacji sumy kwadratów różnic pomiędzy wartościami zmiennych zależnych, a wartościami przewidywanymi określanej wzorem:

$$S(a,b) = \sum_{i=1}^{n} [y_i - y(x_i)]^2 = \sum_{i=1}^{n} [y_i - b - ax_i]^2 \tag{2}$$

Gdzie:

- n → liczba punktów *P(x,y)*.

Minimum funkcji wielu zmiennych występuje w punkcie, w którym wszystkie pochodne cząstkowe względem tych zmiennych są równe zeru.

$$\frac{\partial S(a,b)}{\partial a} = 0 \tag{3}$$

$$\frac{\partial S(a,b)}{\partial b} = 0 \tag{4}$$

Po wyliczeniu pochodnych możemy wyznaczyć parametry *a* oraz *b*.

$$a = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} {x_i}^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \tag{5}$$

$$b = \frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} {x_i}^2 - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i x_i}{n \sum_{i=1}^{n} {x_i}^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \tag{6}$$

W analizie regresji również często używany jest współczynnik korelacji *R* do oceny dopasowania między zmiennymi x i y. Jeżeli wynik korelacji jest bliski 1 lub -1 to wielkości są dobrze skorelowane. Natomiast gdy współczynnik jest równy 1 to wielkości są całkowicie skorelowane, gdy 0 to dopasowania nie ma.

Współczynnik korelacji wyraża się poniższym wzorem.

$$R = \frac{n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n \sum_{i=1}^{n} {x_i}^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \sqrt{n \sum_{i=1}^{n} {y_i}^2 - \left(\sum_{i=1}^{n} y_i\right)^2}} \tag{7}$$

3. Implementacja

W ramach ćwiczeń zaimplementowano aproksymacje metodą najmniejszych kwadratów w języku C++. Kluczowym elementem implementacji jest funkcja „*linearApproximation*" do której szczegółowo odniosę się poniżej.

Napisany program wczytuje dane z pliku „*NM3.txt*" do zmiennej (*read*) oraz deklaruję ilość punktów dla których będzie wykonywać zadanie (*size*).

```
int size;
fstream read("NM3.txt");
```

Następnie sprawdzamy, czy plik został poprawnie otwarty.

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

Gdy plik nie zostanie poprawnie otwarty informuje, o wystąpieniu konfliktu poprzez wyświetlony w terminalu komunikat.

```
else
   cout << "Failed to open the file :<" << endl;
```

Natomiast po pozytywnym otwarciu pliku wczytywana jest liczba punktów funkcji, dla których będziemy znajdować najbardziej dopasowaną funkcję liniową do wcześniej zadeklarowanej tablicy dwuwymiarowej (*points*). Po przypisaniu punktów do tablicy wypisuję dane z niej przy pomocy pętli for.

```
read >> size;   // Liczba punktów
cout << "The number of points for which we complete the task: " << size << endl;

double** points = new double* [size];   // Tablica wskaźników

// Alokacja pamięci dla każdej tablicy wewnętrznej
for (int i = 0; i < size; i++) {
    points[i] = new double[2];
}

// Odczytanie danych do tablicy dwuwymiarowej
for (int i = 0; i < size; i++)
   read >> points[i][0] >> points[i][1];

cout << "Print points" << endl;
for (int i = 0; i < size; i++)
    cout << "X: " << points[i][0] << " Y: " << points[i][1] << endl;
```

W kolejnym kroku deklarowane są zmienne „*a*", „*b*", „*R*" do których będzie przekazywany znaleziony współczynnik kierunkowy, wyraz wolny funkcji oraz współczynnik korelacji.

```
double a, b, R;
```

Następnie przejdę do omówienia głównej funkcji, która jest wywoływana poniżej deklaracji - „`linearApproximation()`". Wyznacza ona parametry *a*, *b* opisane wzorami (5) i (6) oraz współczynnik korelacji (7).

Do funkcji przekazywane są w formie referencji zmienne „*a*", „*b*", „*R*" oraz tablica punktów i ich ilość. Następnie do obliczenia szukanych wartości tworzone są zmienne pomocnicze „*sum_x*", „*sum_y*" – przechowujące sumy wartości x i y, „*sum_x2*" i „*sum_y2*" – przechowujące sumę kwadratów wartości x i y, „*sum_xy*" trzymającą iloczyn wartości x i y. Do podniesienia wartości do potęgi używam funkcji z biblioteki matematycznej „cmath". W pętli for() przechodzącej po wszystkich punktach w tablicy obliczane są odpowiednie sumy oraz poniżej obliczane odpowiednie wartości „a", „b" i „R".

```
void linearApproximation(double **points, double& a, double& b, int size, double &R) {
    double sum_x = 0, sum_y = 0, sum_x2 = 0, sum_xy = 0, sum_y2 = 0;

    // Obliczanie sum potrzebnych do obliczenia współczynników a i b
    for (int i = 0; i < size; ++i) {    //Przejscie przez wszystkie punkty
        sum_x += points[i][0];
        sum_y += points[i][1];
        sum_x2 += pow(points[i][0], 2);
        sum_y2 += pow(points[i][1], 2);
        sum_xy += points[i][1] * points[i][0];
    }

    // Obliczanie współczynników a i b wedlug wzoru
    a = (size * sum_xy - sum_x * sum_y) / (size * sum_x2 - pow(sum_x, 2));
    b = (sum_y * sum_x2 - sum_x * sum_xy) / (size * sum_x2 - pow(sum_x, 2));

   // Oblicznie współczynnika korelacji R, gdzy wynosi -1 lub 1 to wyniki dopasowania jest
bardzo dobry gdzy 0 to przeciwnie
    R = (size * sum_xy - sum_x * sum_y)/(sqrt(size * sum_x2 - pow(sum_x, 2)) * sqrt(size *
sum_y2 - pow(sum_y,2)));
}
```

W dalszej części programu głównego – po wywołaniu funkcji „`linearApproximation()`„ wypisujemy najbardziej dopasowaną funkcję oraz sprawdzamy wynik współczynnika korelacji.

```
if(b > 0)
  cout << "y=" << a << "*x+" << b << endl;
else if (b == 0)
  cout << "y=" << a << "*x" <<  endl;
else
  cout << "y=" << a << "*x" << b << endl;

// Współczynnik korelacji R
cout << "R: " << R << endl;
```

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

```
// Zwolnienie pamięci
        for (int i = 0; i < size; i++) {
            delete[] points[i];
        }
        delete[] points;

read.close();
```

Poniżej przedstawione są zdjęcia całej implementacji algorytmu wraz z odczytem danych z pliku oraz zwolnieniem pamięci.

```cpp
// Funkcja do obliczania aproksymacji liniowej metodą najmniejszych kwadratów
void linearApproximation(double **points, double& a, double& b, int size, double &R) {
    double sum_x = 0, sum_y = 0, sum_x2 = 0, sum_xy = 0, sum_y2 = 0;

    // Obliczanie sum potrzebnych do obliczenia współczynników a i b
    for (int i = 0; i < size; ++i) {    //Przejscie przez wszystkie punkty
        sum_x += points[i][0];
        sum_y += points[i][1];
        sum_x2 += pow(points[i][0], 2);
        sum_y2 += pow(points[i][1], 2);
        sum_xy += points[i][1] * points[i][0];
    }

    // Obliczanie współczynników a i b wedlug wzoru
    a = (size * sum_xy - sum_x * sum_y) / (size * sum_x2 - pow(sum_x, 2));
    b = (sum_y * sum_x2 - sum_x * sum_xy) / (size * sum_x2 - pow(sum_x, 2));

   // Oblicznie współczynnika korelacji R, gdzy wynosi -1 lub 1 to wyniki dopasowania jest bardzo dobry gdzy 0 to przeciwnie
    R = (size * sum_xy - sum_x * sum_y)/(sqrt(size * sum_x2 - pow(sum_x, 2)) * sqrt(size * sum_y2 - pow(sum_y,2)));
}
```

Rysunek 1 Fragment kodu - główna funkcja „linearApproximation()”

```cpp
//Odczyt danych punktów dla których szukamy funkcji
int size;
fstream read("MN3.txt");

if (read.is_open()) {
    read >> size;   // Liczba punktów
    cout << "The number of points for which we complete the task: " << size << endl;

    double** points = new double* [size];   // Tablica wskaźników

    // Alokacja pamięci dla każdej tablicy wewnętrznej
    for (int i = 0; i < size; i++) {
        points[i] = new double[2];
    }

    // Odczytanie danych do tablicy dwuwymiarowej
    for (int i = 0; i < size; i++)
        read >> points[i][0] >> points[i][1];

    cout << "Print points" << endl;
    for (int i = 0; i < size; i++)
        cout << "X: " << points[i][0] << " Y: " << points[i][1] << endl;

    //Deklaracja wartosci a, b, których szukamy
    double a, b, R;

    linearApproximation(points, a, b, size, R);    //Dla szukanej y=ax+b

    if(b > 0)
        cout << "y=" << a << "*x+" << b << endl;
    else if (b == 0)
        cout << "y=" << a << "*x" <<  endl;
    else
        cout << "y=" << a << "*x" << b << endl;

    // Współczynnik korelacji R
   cout << "R: " << R << endl;

    // Zwolnienie pamięci
    for (int i = 0; i < size; i++) {
        delete[] points[i];
    }
    delete[] points;
}
else
    cout << "Failed to open the file :<" << endl;

read.close();
```

Rysunek 2 Fragment implementacji – program główny (main)

4. Testy jednostkowe

Testy programu zostały porównane do wyników przedstawionych przez program „Excel”.

a. Test 1 przeprowadzony dla punktów funkcji f(x)= -3 x+7
   Podane punkty:
   - x=0 y=7
   - x=1 y=4
   - x=2 y=1
   - x=3 y=-2

   Wyniki działania programu:
   - R = -1

- y= -3x + 7

Wyniki programu Excel:

- R = -1
- y= -3x + 7

```
The number of points for which we complete the task: 4
Print points
X: 0 Y: 7
X: 1 Y: 4
X: 2 Y: 1
X: 3 Y: -2
y=-3*x+7
R: -1
Press any key to continue . . .
```

Rysunek 3 Wyniki programu dla testu a

b. Test 2 przeprowadzony dla punktów funkcji f(x)=1.5x-2

Podane punkty:

- x=0 y=-2
- x=4 y=4
- x=-2 y=-5

Wyniki działania programu:

- R=1
- y=1,5x - 2

Wyniki programu Excel:

- R = 1
- y=1,5x - 2

```
The number of points for which we complete the task: 3
Print points
X: 0 Y: -2
X: 4 Y: 4
X: -2 Y: -5
y=1.5*x-2
R: 1
Press any key to continue . . .
```

Rysunek 4 Wyniki programu dla testu b

c. Test 3 przeprowadzony dla 5 punktów.

Podane punkty:

- x=1 y=5
- x=2 y=7
- x=3 y=9
- x=4 y=11
- x=5 y=13

Wyniki działania programu:

- R=1
- y=2x + 3

Wyniki programu Excel:

- R = 1
- y= 2x + 3

```
The number of points for which we complete the task: 5
Print points
X: 1 Y: 5
X: 2 Y: 7
X: 3 Y: 9
X: 4 Y: 11
X: 5 Y: 13
y=2*x+3
R: 1
Press any key to continue . . .
```

Rysunek 5 Wyniki dla testu c

d. Test 3 przeprowadzony dla 15 punktów.
Podane punkty:

- x=1 y=2
- x=2 y=3
- x=3 y=6
- x=4 y=7
- x=5 y=10
- x=6 y=12
- x=7 y=13
- x=8 y=13
- x=9 y=18
- x=10 y=20
- x=11 y=22
- x=12 y=24
- x=13 y=26
- x=14 y=26
- x=15 y=29

Wyniki działania programu:

- R=0,99
- y=1,97x-0,37

Wyniki programu Excel:

- R=0,99
- y=1,97x-0,37

```
The number of points for which we complete the task: 15
Print points
X: 1 Y: 2
X: 2 Y: 3
X: 3 Y: 6
X: 4 Y: 7
X: 5 Y: 10
X: 6 Y: 12
X: 7 Y: 13
X: 8 Y: 13
X: 9 Y: 18
X: 10 Y: 20
X: 11 Y: 22
X: 12 Y: 24
X: 13 Y: 26
X: 14 Y: 26
X: 15 Y: 29
y=1.97143*x-0.371429
R: 0.994816
Press any key to continue . . .
```

Rysunek 6 Wyniki dla testu d

e. Test 5 przeprowadzony dla 10 punktów.
Podane punkty:

- x=-0,5 y=2
- x=-0,5 y=3
- x=1,5 y=6
- x=3,6 y=7
- x=3,2 y=7,1
- x=5,8 y=8
- x=4,2 y=6
- x=8,1 y=9
- x=9 y=15
- x=9,8 y=11

Wyniki działania programu:

- R=0,91
- y=0,917x+3,356

Wyniki programu Excel:

- R=0,91
- y=0,917x+3,356

```
The number of points for which we complete the task: 10
Print points
X: -0.5 Y: 2
X: -0.5 Y: 3
X: 1.5 Y: 6
X: 3.6 Y: 7
X: 3.2 Y: 7.1
X: 5.8 Y: 8
X: 4.2 Y: 6
X: 8.1 Y: 9
X: 9 Y: 15
X: 9.8 Y: 11
y=0.917135*x+3.35626
R: 0.91053
Press any key to continue . . .
```

Rysunek 7 Wyniki dla testu e

5. Opracowanie wyników

Zestawienie zawarte w poniższej tabeli przedstawia wszystkie wyniki wyznaczonych współczynników funkcji oraz współczynnika korelacji.

Tabela 1. Opracowanie uzyskanych wyników

Przeprowadzony test	Excel			Wyniki programu		
	a	b	R	a	b	R
a	-3	7	-1	-3	7	-1
b	1,5	-2	1	1,5	-2	1
c	2	3	1	2	3	1
d	1,97	0,37	0,99	1,97	0,37	0,99
e	0,917	3,356	0,91	0,917	3,356	0,91

Opracowanie wyników zachodzi przy założeniu, że Excel dostarcza poprawne dane, a program jest porównywany i analizowany względem tych wyników, stanowi to kluczowy etap oceny skuteczności implementacji i poprawności działania algorytmu.

Każdy przeprowadzony test obejmował różne funkcje oraz zestawy danych, co pozwoliło na zbadanie skuteczności implementacji aproksymacji metodą najmniejszych kwadratów. Przy testach a, b oraz c R wyniosło -1 lub 1 co oznacza bardzo dobre dopasowanie funkcji do zbioru punktów. Natomiast przy d, e wyniki odpowiednio wyniosły R=0,99 i 0,91 dla napisanego programu i Excela co również jest bardzo dobrym dopasowaniem. Analiza tych wyników zwróconych dla a, b i R pozwala stwierdzić, że implementacja aproksymacji metodą

najmniejszych kwadratów w języku C++ jest skuteczna i daje zgodne rezultaty z programem Excel, został on poprawnie zaimplementowany. Warto jednak zauważyć, że pomimo wysokiej skuteczności, każda aproksymacja wymaga starannego przemyślenia i analizy, aby zagwarantować odpowiednie dopasowanie modelu do danych oraz interpretację uzyskanych wyników.

6. Wnioski

Badanie aproksymacji za pomocą metody najmniejszych kwadratów wskazuje na jej efektywność w dopasowywaniu funkcji do zestawu danych. Ta technika umożliwia znalezienie optymalnego przybliżenia funkcji. Skuteczność programu w odwzorowywaniu rzeczywistości zależy od precyzji obliczeń i dokładności uzyskiwanych wyników. Im dokładniejsze są wyniki programu, tym lepiej odzwierciedlają one rzeczywiste związki między zmiennymi.

7. Źródła

- Prezentacja autorstwa dr hab. inż. Marcina Hojnego „Aproksymacja – metoda najmniejszych kwadratów”.