Wydział WIMiIP	Imię i Nazwisko Mateusz Witkowski	Rok II	Grupa 4
Kierunek IS	Temat Interpolacja Newton'a	Prowadzący dr hab. inż. Hojny Marcin, prof. AGH	
Data ćwiczenia 19.03.20	Data oddania 26.03.20	Data zaliczenia	Ocena

1. Cel ćwiczenia:

Celem ćwiczenia było napisanie implementacji metody numerycznej – interpolacji Newton'a pozwalającej znalezienie wartości funkcji w dowolnym punkcie.

2. Wprowadzenie do metody:

Interpolacja Newton'a (lub też interpolacja wielomianowa) jest to metoda numeryczna wykorzystywana do przybliżania przebiegu funkcji. Wybieranych jest n+1 punktów zwanych węzłami interpolacji, należących do dziedziny danej funkcji. W punktach tych funkcja(wielomian) przyjmuje wartości takie same jak przybliżana funkcja.
Funkcję interpolującą w postaci wielomianu stopnia n wyznacza się w postaci:

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

Gdzie:
x – argument dla którego wartość chcemy znaleźć
$\mathbf{y_0}$ - wartość funkcji odpowiadająca $\mathbf{x_0}$

Przy wyliczaniu tego wielomianu musimy zdefiniować wyrażenia zwane ilorazami różnicowymi:

Rzędu pierwszego: $$[x_i, x_j] = \frac{y_j - y_i}{x_j - x_i}$$

Drugiego rzędu: $$[x_i, x_j, x_k] = \frac{[x_j, x_k] - [x_i, x_j]}{x_k - x_i}$$

Rzędów wyższych:

$$[x_i, x_{i+1}, \dots, x_{i+k}] = \frac{[x_{i+1}, x_{i+2}, \dots, x_{i+k}] - [x_i, x_{i+1}, \dots, x_{i+k-1}]}{x_{i+k} - x_i}$$

Jak łatwo można zauważy wzory na ilorazy różnicowe kolejnych rzędów są oblicza się z zależności rekurencyjnych, co pozwala na łatwe obliczenie ilorazu k-tego rzędu, znając dwa ilorazy rzędu o jeden niższego.

3. Kod programu

Funkcja odpowiadająca za wczytywanie danych z pliku, i zapisanie ich do wektora.

```
//Funkcja pobierjąca dane z pliku tekstowego i zapisujące je do wektora.
int pobranie_z_pliku(vector <double>* data, string fileName) {
    string linia;
    fstream plik;

    plik.open(fileName, ios::in);
    if (plik.good() == true)
    {
        while (!plik.eof())
        {
            getline(plik, linia, ',');      //zapisuje słowa odzielane przecinkami
            double d = atof(linia.c_str());
            data->push_back(d);             //zapis słowa parsowanego na typ double do wektora
        }
        plik.close();
    }
    if (data->size() % 2 == 1) {    //zabezpiecznie w razie źle wypełnionego pliku tekstowego
        cout << "bledne dane w pliku" << endl;
        return -1;
    }
    return data->size();    //funkcja zwraca wielkosc wektora potrzebna do stworzenia tablic przechowujących X i Y
}
```

Funkcja main odpowiadająca za wywołanie wszystkich funkcji w programie.

```
int main() {
    ///////////SPRAWDZZANIE POPRAWNOSCI PLIKU///////////
    int wielkosc_wektora = pobranie_z_pliku(&dane, nazwaPliku);
    if (wielkosc_wektora == -1) //program zostanie przerwany jesli plik tekstowy był błędny
    {
        return -1;
    }
    ///////////////////////

    ///////////WYPEŁNIANIE TABLIC///////////
    int liczba_punktow = wielkosc_wektora / 2; //wielkosc wektora podzielona przez 2 powinna nam dać iloś punktów zapisanych w pliku
    double* X = new double[liczba_punktow];     //tablica współrzędnych x
    double* Y = new double[liczba_punktow];     //tablica współrzędnych y
    int j = 0;
    for (size_t i = 0; i < dane.size(); i++)    //wypełanienie tablic
    {
        if (i % 2 == 0) {
            X[j] = dane[i];
        }
        else {
            Y[j] = dane[i];
            j++;
        }
    }
    ///////////////////////
```

Ciąg dalszy funkcji main odpowiedzialny za wyświetlenie menu.

```
    double szukana;
    char wybor;
    ////////////MENU////////////
    for (;;)
    {
        cout << endl;
        cout << "MENU GLOWNE" << endl;
        cout << "-----------------" << endl;
        cout << "1. Interpolacja Lagrange'a" << endl;
        cout << "2. Interpolacja Newtona'a" << endl;
        cout << "3. Koniec programu" << endl;

        cout << endl;
        cin >> wybor;

        switch (wybor)      //switch pozwalajacy wybrac funkcje z menu
        {
        case '1':
            cout << "Interpolacja Lagrange'a dla punktow" << endl;
            for (size_t i = 0; i < liczba_punktow; i++) //wypisanie zawartosci tablic
            {
                    cout << "f(" << X[i] << ") = " << Y[i] <<endl;
            }
            cout << "Podaj wartosc szukanego x: ";
            cin >> szukana;
            cout << endl;
            cout << "Wartosc dla podanego x to: " << lagrange(liczba_punktow, X, Y, szukana) << endl;
            break;

        case '2':
            cout << "Interpolacja Newtona'a dla punktow" << endl;
            for (size_t i = 0; i < liczba_punktow; i++) //wypisanie zawartosci tablic
            {
                cout << "f(" << X[i] << ") = " << Y[i] << endl;
            }
            cout << "Podaj wartosc szukanego x: ";
            cin >> szukana;
            cout << endl;
            cout << "Wartosc dla podanego x to: " << newton(liczba_punktow, X, Y, szukana) << endl;
            break;
        case '3':
            exit(0);
            break;

        default: cout << "Nie ma takiej opcji w menu!";
        }
        getchar(); getchar();
        system("cls");
    }
    ////////////////////////


    return 0;
}
```

Funkcja realizująca interpolacje Newtona.

```cpp
double newton(int liczba_punktow, double* X, double* Y, double szukana) {
    //na poczatku do wyniku dopisujemy Y[0], czyli f(x[0])  - tak jak w rownaniu W[n](x) = f(x[0]) + ...
    double wynik = Y[0];

    //tablica ta bedzie przechowywac ilorazy roznicowe kolejnych rzedow.
    double* ilorazy_roznicowe = new double[liczba_punktow];
    ilorazy_roznicowe[0] = Y[0];

    for (size_t i = 1; i < liczba_punktow; i++)
    {
        ilorazy_roznicowe[i] = Y[i];
        double iloczyn_roznic = 1.0;
        //zmienna potrzebna do wyliczania kolejnych iloczynow
        //(x-x[0])(x-x[1])(x-x[2])*...*(x-x[n-1]) -> x to nasza szukana w tym przypadku

        for (size_t j = 0; j < i; j++)
        {
            //obliczanie ilorazow kolejnych rzedow
            ilorazy_roznicowe[i] = (ilorazy_roznicowe[i] - ilorazy_roznicowe[j]) / (X[i] - X[j]);
            iloczyn_roznic = iloczyn_roznic*(szukana - X[j]);   //obliczanie kolejnych iloczynow

        }
        cout << "iloraz roznicowy rzedu " << i << " rowny jest " << ilorazy_roznicowe[i] << endl;
        //wyswietlenie ilorazow roznicowych kolejnych rzedow.
        wynik += ilorazy_roznicowe[i] * iloczyn_roznic;
        //powiekszanie wyniku o kolejne "bloki":
        //-> [x[0],x[1]](x-x[0])+...+ [x[0],x[1],...,x[n]](x-x[0])(x-x[1])(x-x[2])*...*(x-x[n-1])

    }
    delete[] ilorazy_roznicowe;     //zwalniamy miejsce w pamieci
    return wynik;       //zwracamy wynik
}
```

4. Testy poprawności działania algorytmu

Najpierw by sprawdzić czy program działa porównano wyniki z tym przedstawionymi w przykładzie na stronie http://galaxy.agh.edu.pl/~mhojny/repozytoria/mn/InterpolacjaN.pdf

Szukana wartość dla x = 1

Wynik w przykładzie: 44

Plik tekstowy:

```
punkty.txt    Source.cpp
1  -2, -1,
2  -1, 0,
3  0, 5,
4  2, 99,
5  4, -55
```

Menu wyboru:

```
MENU GLOWNE
-----------------
1. Interpolacja Lagrange'a
2. Interpolacja Newtona'a
3. Koniec programu
```

Wynik:

```
D:\STUDIA\IV_Semestr\Metody\zajęcia3\InterpolacjaNewtona\Debug\InterpolacjaNewtona.exe

MENU GLOWNE
-----------------
1. Interpolacja Lagrange'a
2. Interpolacja Newtona'a
3. Koniec programu

2
Interpolacja Newtona'a dla punktow
f(-2) = -1
f(-1) = 0
f(0) = 5
f(2) = 99
f(4) = -55
Podaj wartosc szukanego x: 1

iloraz roznicowy rzedu 1 rowny jest 1
iloraz roznicowy rzedu 2 rowny jest 2
iloraz roznicowy rzedu 3 rowny jest 3
iloraz roznicowy rzedu 4 rowny jest -2
Wartosc dla podanego x to: 44
```

Następnie wykonano test przy pomocy programu WolframAlpha. Do programu wprowadzono losowo wybrane punkty w celu uzyskania wykresu oraz wzoru funkcji. Dzięki czemu można było łatwo podstawić szukane wartości i porównać wyniki z Wolframa i stworzonego programu.

Test1

Szukana wartość dla x=2

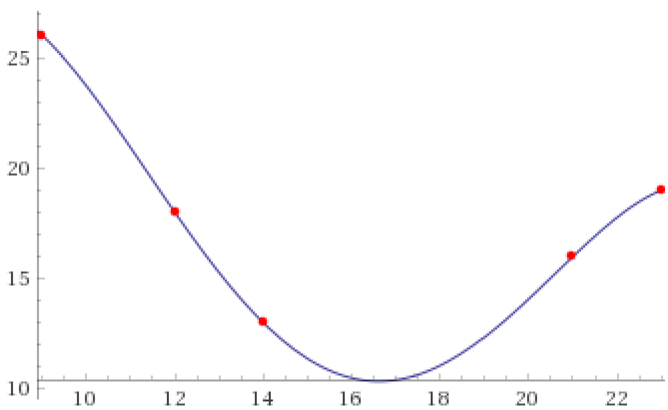
Wolfram

interpolating polynomial	{{9, 26}, {23, 19}, {14, 13}, {12, 18}, {21, 16}}

Interpolating polynomial:

$$-\frac{32 x^4}{10395}+\frac{409 x^3}{2079}-\frac{17957 x^2}{4158}+\frac{51365 x}{1386}-\frac{4452}{55}$$

Plot of the interpolating polynomial:



					Wynik
- (32 x^4)/10395	+ (409 x^3)/2079	- (17957 x^2)/4158	+ (51365 x)/1386	-80,9455	-22,5758
-0,049254	1,5738	-17,2747	74,11977	-80,9455	

Program:

D:\STUDIA\IV_Semestr\Metody\zajęcia3\InterpolacjaNewtona\Debug\InterpolacjaNewtona.exe

```
MENU GLOWNE
-----------------
1. Interpolacja Lagrange'a
2. Interpolacja Newtona'a
3. Koniec programu

2
Interpolacja Newtona'a dla punktow
f(9) = 26
f(12) = 18
f(14) = 13
f(21) = 16
f(23) = 19
Podaj wartosc szukanego x: 2

iloraz roznicowy rzedu 1 rowny jest -2.66667
iloraz roznicowy rzedu 2 rowny jest 0.0333333
iloraz roznicowy rzedu 3 rowny jest 0.0243386
iloraz roznicowy rzedu 4 rowny jest -0.0030784
Wartosc dla podanego x to: -22.5758
```

Test2

Szukana wartość dla x=4

Wolfram

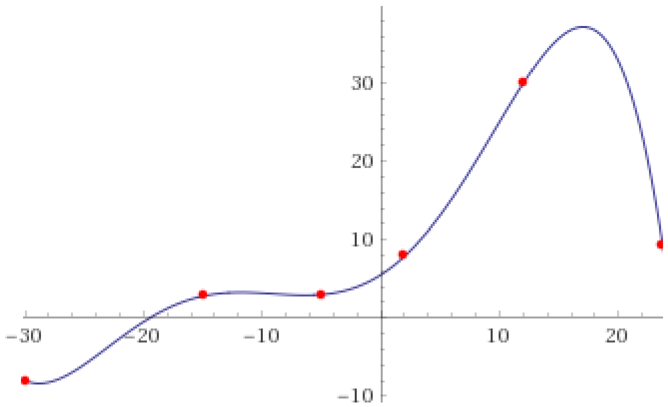
interpolating polynomial	{{23.7, 9.4}, {−15, 2.8}, {12, 30}, {2, 7.9}, {−5, 3}, {−30, −8}}

Interpolating polynomial:

Enlarge

$$-5.2868\times10^{-6}\,x^5-0.000197866\,x^4+0.00176832\,x^3+0.105635\,x^2+0.968476\,x+5.5297$$

Plot of the interpolating polynomial:



						Wynik
- 5.2868×10^-6 x^5	- 0.000197866 x^4	+ 0.00176832 x^3	+ 0.105635 x^2	+ 0.968476 x	5.5297 - wolny wyraz	11,15087
-0,00541	-0,05065	0,11317	1,69016	3,8739	5,5297	

Program:

```
D:\STUDIA\IV_Semestr\Metody\zajęcia3\InterpolacjaNewtona\Debug\InterpolacjaNewtona.exe

MENU GLOWNE
-----------------
1. Interpolacja Lagrange'a
2. Interpolacja Newtona'a
3. Koniec programu

2
Interpolacja Newtona'a dla punktow
f(23.7) = 9.4
f(-15) = 2.8
f(12) = 30
f(2) = 7.9
f(-5) = 3
f(-30) = -8
Podaj wartosc szukanego x: 4

iloraz roznicowy rzedu 1 rowny jest 0.170543
iloraz roznicowy rzedu 2 rowny jest -0.0715269
iloraz roznicowy rzedu 3 rowny jest -0.00655611
iloraz roznicowy rzedu 4 rowny jest -0.000291442
iloraz roznicowy rzedu 5 rowny jest -5.2868e-06
Wartosc dla podanego x to: 11.1509
```

5.Wnioski

Interpolacja Newtona’a jest bardzo dobrą metodą interpolacyjną, umożliwia znalezienie jednoznacznego rozwiązania, gdy posiadamy wiedze o należących do funkcji punktach i ich wartościach. Większa ilość znanych punktów zapewnia większa dokładność. Jej główną zaletą w stosunku do interpolacji Lagrange’a jest to, że świetnie sprawdza się w sytuacji zmieniającej się liczby węzłów, gdyż nie musimy powtarzać obliczeń od początku, za każdym razem gdy dodajemy nowy punkt. Wystarczy jedynie zmodyfikować wyznaczony wcześniej wielomian. Największe koszty obliczeniowe tego algorytmu Newtona wiążą się z wyznaczaniem kolejnych ilorazów różnicowych. Testy potwierdziły poprawność działania programu zarówno względem wyników z Wolframa jak i tych obliczonych w tradycyjny sposób.