Temat: Wyznaczanie pierwiastków równania nieliniowego metodą Newtona			
Wykonał: Marcin Fabrykowski	Wydział: FiIS	Kierunek Inf. Stos.	Grupa: grupa 3

1. Wstęp Wyznaczanie pierwiastków równania nieliniowego metodą Newtona na wyznaczeniu przedziału izolacji danego pierwiastka. W naszym przypadku wykonujemy to poprzez odczytanie przedziałów z wykresu. Taki przedział oznaczamy poprzez $< a, b >$. Następnie wyznaczamy styczną do wykresu w punkcie b. Równanie opisujące tą styczną:

   $$y - f(b) = f'(b)(x - b)$$

   Przyjmujemy $y = 0$ z racji tego, że szukamy x w którym styczna przecina oś OX.

   $$x_1 = b - \frac{f(b)}{f'(b)}$$

   Następnie sprawdzamy czy $b - x_1 < \varepsilon$. Jeśli powyższy warunek jest spełniony, wtedy przyjmujemy x_1 jako pierwiastek równania. W przeciwnym wypadku przyjmujemy nowe $b = x_1$ i powtarzamy procedurę. Powyższy schemat powtarzamy dla wszystkich pierwiastków.

2. Wykonanie ćwiczenia. Badamy funkcję:

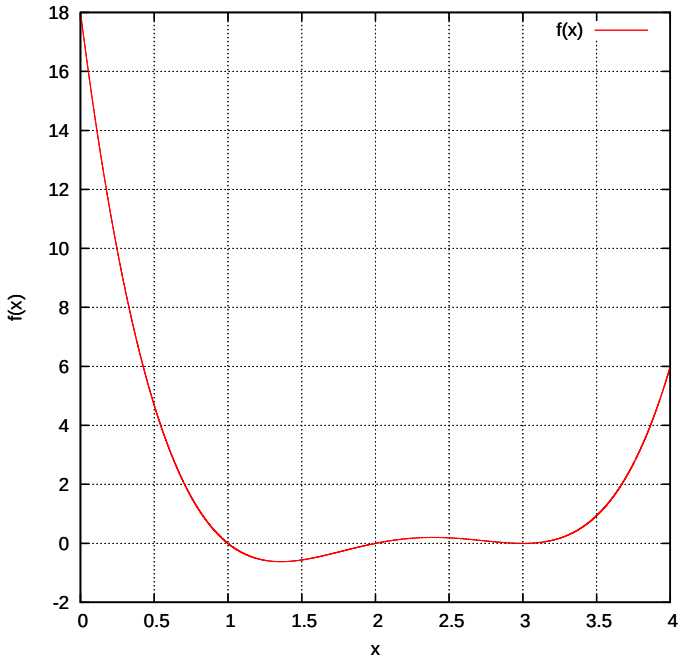
   $$f(x) = x^4 - 9x^3 + 29x^2 - 39x + 18$$

   Jej wykres przedstawia rys. 1
   Zauważamy że przybliżone pierwiastki równania wynoszą: 1,2 oraz 3 dlatego przedziały izolacji wybieramy ± 0.2
   Do wyliczenia pierwiastków wykorzystujemy własnoręcznie napisany program bazujący na algorytmie podanym we wstępie:

   Listing 1: main.cpp

```
#include <iostream>
#include <vector>
#include <cmath>
#include <string>
#include <fstream>
```

Rysunek 1: Wykres f(x)




```
#include <iomanip>
using namespace std;
vector<float> rozniczkuj(vector<float> wsp);
float oblicz(vector<float>,float x1);
void zapisz(string path,vector<float>,float a, float
    b,float krok);
float znajdz(string path,vector<float> wsp,float a, float
    b,float eps);
int main()
{
        int stopien=5;
        vector<float> wsp;
        wsp.push_back(1);
        wsp.push_back(-9);
        wsp.push_back(29);
        wsp.push_back(-39);
        wsp.push_back(18);

        cout<<"Mamy "<<stopien-1<<" pierwiastki/ow"<<endl;

        rozniczkuj(wsp);
        cout<<"rysowanie funkcji"<<endl;
        zapisz("f.dat",wsp,0,4,0.0001);
        cout<<"wyznaczanie pierwszego pierwiastka..."<<endl;
```

```
        cout<<znajdz("1.dat",wsp,0.8,1.2,0.000001)<<endl;
        cout<<"wyznaczanie drugiego pierwiastka..."<<endl;
        cout<<znajdz("2.dat",wsp,01.8,2.2,0.000001)<<endl;
        cout<<"wyznaczanie trzeciego pierwiastka..."<<endl;
        cout<<znajdz("3.dat",wsp,2.8,3.2,0.000001)<<endl;
};
vector<float> rozniczkuj(vector<float> wsp)
{
        vector<float> wynik;
        unsigned int x;
        for(x=0;x<wsp.size()-1;x++)
        {
        //      cout<<"Przed: "<<wsp[x]<<endl;
                wynik.push_back(wsp[x]*(wsp.size()-x-1));
        //      cout<<"Po: "<<wynik[x]<<endl;
        };
        return wynik;
};
float oblicz(vector<float> wsp,float x1)
{
        float wynik=0;
        unsigned int x;
        for(x=0;x<wsp.size();x++)
        {
                wynik+=wsp[x]*pow(x1,wsp.size()-x-1);
        };
        return wynik;
};
void zapisz(string path,vector<float> wsp,float a, float
    b,float krok)
{
        ofstream plik;
        plik.open(path.c_str());
        float x;
        for(x=a;x<b;x+=krok)
        {
                plik<<x<<" "<<oblicz(wsp,x)<<endl;
        };
        plik.close();
};
float znajdz(string path,vector<float> wsp,float a, float
    b,float eps)
{
        ofstream plik;
        plik.open(path.c_str());
        int x=0;
        while(true)
        {
                float f=oblicz(wsp,b);
```

```
                float fp=oblicz(rozniczkuj(wsp),b);
                float x1=b-f/fp;
//              float tmp=oblicz(wsp,x1);
                float tmp=fabs(b-x1);
                plik<<setw(8)<<x++<<"\t";
                plik<<setw(8)<<x1<<"\t";
                plik<<setw(8)<<f<<"\t";
                plik<<setw(8)<<fp<<"\t";
                plik<<setw(8)<<tmp<<endl;
                b=x1;
                if(tmp<eps)
                        break;
        };
        plik.close();
        return b;
};
```

3. Wnioski Dużą zaletą tej metody jest jest szybkość. Mianowicie liczba kroków potrzebnych do znalezienia pierwiastków jest niewielka, co pokazują dane wyjściowe:

   (a) dla pierwiastka pierwszego:

```
0 0.821054 -0.518398   -1.368 0.378946
1 0.957028  1.00163 -7.36638 0.135974
2 0.996697 0.187061 -4.71557 0.0396689
3 0.999978 0.0133018 -4.05302 0.00328195
4        1 8.58307e-05 -4.00034 2.14577e-05
5        1        0       -4        0
```

   (b) dla pierwiastka drugiego:

```
0  1.89999 0.153605 0.511997  0.30001
1   1.9934 -0.108913  1.16601 0.0934068
2  1.99996 -0.00664711  1.01307 0.00656128
3        2 -4.3869e-05  1.00009 4.37498e-05
4        2 1.90735e-06        1 1.90735e-06
5        2        0        1        0
```

   (c) dla pierwiastka trzeciego

```
0  3.11142 0.105593    1.192 0.0885847
1  3.05965 0.0291367 0.562908 0.0517609
2  3.03105 0.00776482   0.2715 0.0285997
3   3.0159 0.00201607 0.133022 0.0151558
4  3.00794 0.000524521 0.0658913 0.00796032
```

```
5  3.00387 0.000131607 0.0323257 0.00407124
6  3.00203 2.86102e-05 0.0155983 0.00183415
7  3.00063 1.14441e-05 0.00817108 0.00140047
8  2.99913 3.8147e-06 0.00253677 0.00150371
9  2.99913         0 -0.00347137         0
```

Jednakże, wymagane jest podanie przedziałów izolacji pierwiastków, co nie jest ani wygodne, ani łatwe do wyznaczenia w sposób automatyczny