Metody Numeryczne

Kacper Marchlewicz

Projekt 3 Zadanie 3.22

1.Proszę znaleźć wszystkie zera funkcji

$$f(x) = 2\sin x - e^x + 5x - 1$$

W przedziale [-5,5], używając dla każdego zera programu z implementacją:

a) metody regula falsi

b) metody Newtona

Wymagane metody są metodami iteracyjnymi wymagającymi oszacowania przedziału izolacji pierwiastka, czyli przedziału w którym znajduje się rozwiązanie. W tym celu stworzyłem program w Matlabie, oparty na zawartym w podręczniku.

```
function [a,b]=comp(f,start,max)
x1=start;
x2=start+0.05;
for j=1:1000
    if f(x1)*f(x2)<0
        a=x1;
        b=x2;
    elseif (abs(f(x1))<abs(f(x2)))
        x1=x1+1.001*(x1-x2);
    elseif x2+1.001*(x2-x1)<max
        x2=x2+1.001*(x2-x1);
    end
end
```

Mając oszacowany przedział pierwiastka, można przystąpić do liczenia jego wartości. Wyznacznikiem szybkości metod jest rząd metody iteracyjnej, czyli największa liczba p, większa lub równa 1, taka, że:

$$\lim_{n\to\infty} \frac{|x_{n+1} - \alpha|}{|x_n - \alpha|^p} = k < \infty$$

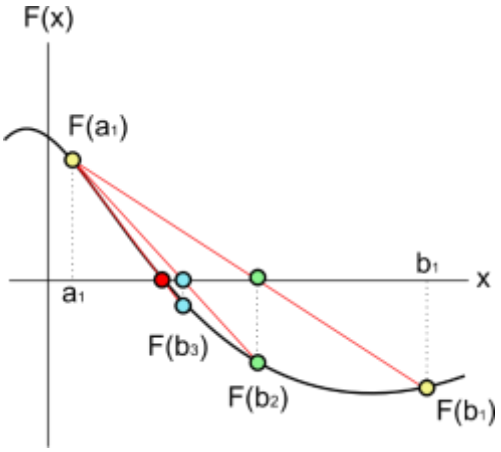
gdzie k jest nazywane współczynnikiem lub ilorazem zbieżności. Im wyższy jest rząd metody, tym metoda zbiega szybciej, potrzebujemy mniej iteracji do jej wykonania. α to pierwiastek (którym może być jeden z krańców przedziału izolacji). Metody iteracyjne dla problemów nieliniowych są zwykle zbieżne tylko lokalnie

Metoda regula falsi

Metoda regula falsi (fałszywa reguła), zwana też metodą false position, jest bardzo podobna do metody bisekcji – różnica polega na tym, że aktualny przedział $[a_n, b_n]$ izolacji pierwiastka α dzielony jest nie na dwa równe, ale na dwa najczęściej nierówne podprzedziały, prostą (sieczną) łączącą na płaszczyźnie (f, x) punkty (f(a_n), a_n) i (f(b_n), b_n), przecinającą oś rzędnych w punkcie oznaczonym jako c_n, który liczony jest ze wzoru:

$$c_n = \frac{a_n f(b_n) - b_n f(a_n)}{f(b_n) - f(a_n)}$$

Wybór następnego przedziału izolacji pierwiastka w metodzie regula falsi jest dokonywany tak samo jak w metodzie bisekcji, przez wyznaczanie iloczynów wartości funkcji na krańcach podprzedziałów i wybór tego podprzedziału, któremu odpowiada iloczyn ujemny.



Metoda Newtona (stycznych)

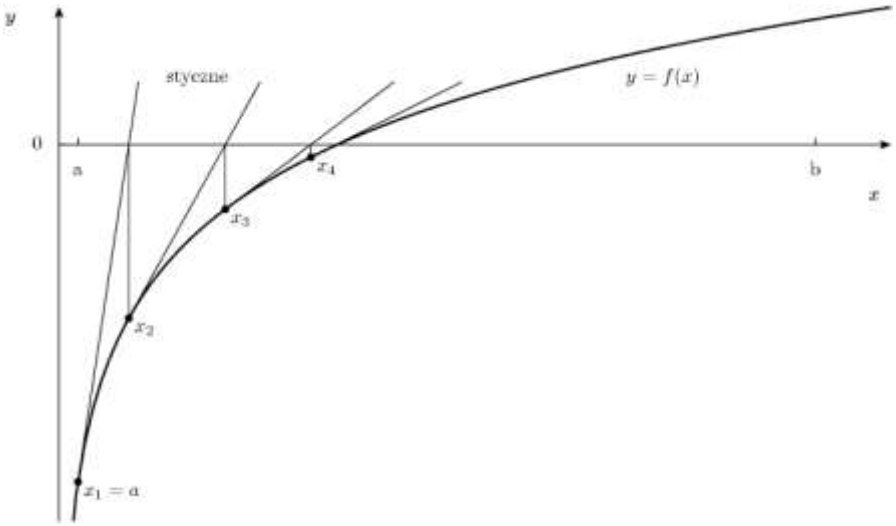
Metoda Newtona, zwana też metodą stycznych, zakłada aproksymacje funkcji jej liniowym przybliżeniem wynikającym z uciętego rozwinięcia w szereg Taylora w aktualnym punkcie x_n (aktualnym przybliżeniu pierwiastka)

$$f(x) \approx f(x_n) + f'(x_n)(x - x_n)$$

co prowadzi do zależności iteracyjnej

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Metoda Newtona jest również zbieżna kwadratowo, posiada rząd równy 2. Metoda jest efektywna, kiedy w otoczeniu pierwiastka funkcja jest stroma, natomiast kiedy w jego otoczeniu jest „płaska” – metoda jest nieskuteczna



Implementacja metody regula falsi w Matlab

```
function [X0,n] = falsi_solver(f,a,b,dokl)
n=0;
X0=[];
x0=0;
for i=1:1000
    n=n+1;
    c=((a*f(b)-b*f(a))/(f(b)-f(a)));
    if abs(f(c))<dokl
        if (a<c) && (c<b)
            old=x0;
            x0=c;
            if abs(x0-old)>dokl
                X0(end+1)=x0;
            end
            if abs(x0-old)<dokl
                break
            end
         end
     elseif f(a)*f(c)>0
        a=c;
     elseif f(b)*f(c)>0
        b=c;
    end
end
end
```

Funkcja wywołująca solver

```
f=@(x) (2*sin(x)) - exp(x) + (5*x)-1;
fplot(f,[-6 6]);
grid on

dokl=10^-6;

[startuj1,koncz1]=comp(f,-5,5);
[zerowe1,iteracje1]=falsi_solver(f,startuj1,koncz1,dokl);

fn=@(x)f(x)./(x-zerowe1);
[startuj2,koncz2]=comp(fn,koncz1,5);
[zerowe2,iteracje2]=falsi_solver(f,startuj2,koncz2,dokl);
```

Implementacja metody Newtona w Matlab

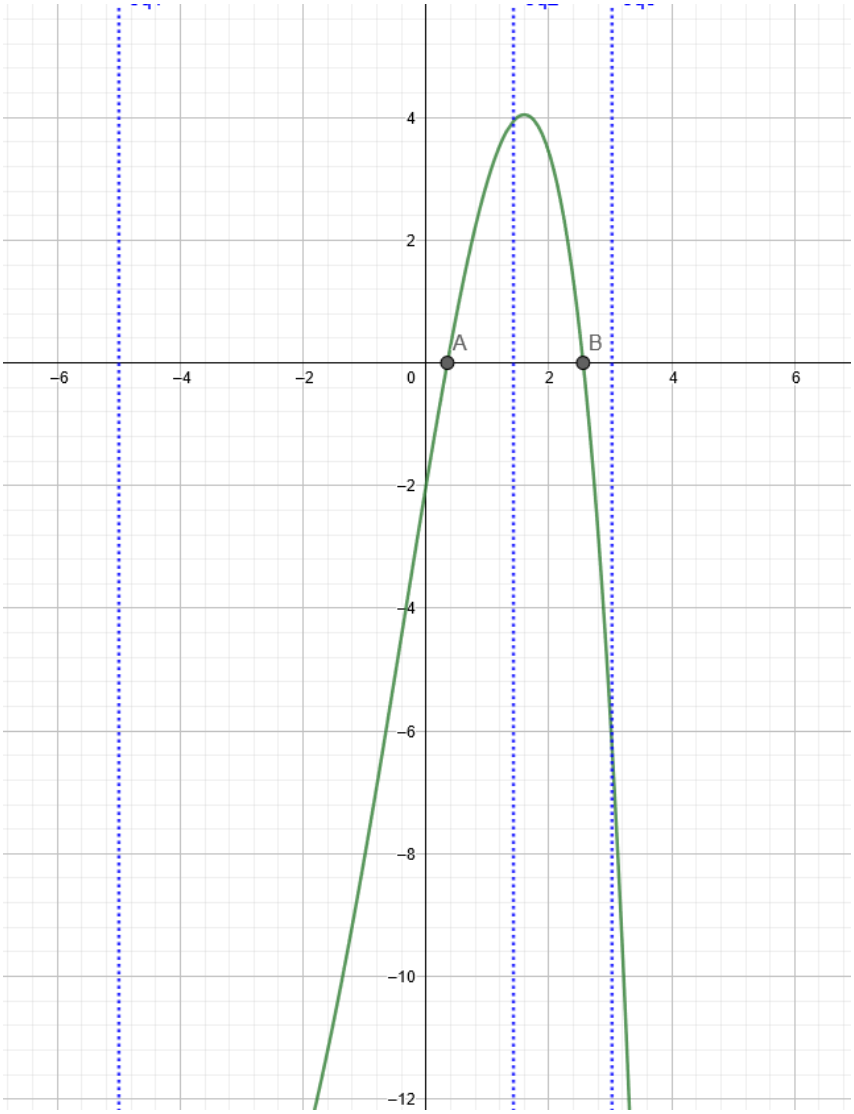
```
function [X0,n] = newton_solver(f,df,x1,x2,dokl)
n=0;
X0=[];
x0=0;
for i=1:1000
    while abs(f(x1))>dokl
        n=n+1;
        x1=x1-(f(x1)/df(x1));
    end
    old=x0;
    x0=x1;
    if abs(x0-old)<dokl
        break
    end
    if (x0>=x1 && x0<=x2)
```

```
        X0(end+1)=x0;
    end
end
end
```

Funkcja wywołująca solver

```
f=@(x) (2*sin(x)) - exp(x) + (5*x)-1;
df=@(x) (2*cos(x))- exp(x) +5;
fplot(f,[-6 6]);
grid on

dokl=10^-6;
[startuj1,koncz1]=comp(f,-5,5);
[zerowe1,iteracje1]=newton_solver(f,df,startuj1,koncz1,dokl);


fn=@(x)f(x)./(x-zerowe1);
[startuj2,koncz2]=comp(fn,koncz1,5);
[zerowe2,iteracje2]=newton_solver(f,df,startuj2+0.8,koncz2,dokl);
```

Wykres funkcji



Wyniki

Metoda regula falsi			Metoda Newtona		
Wyniki:	0,3469	2,5561	Wyniki:	0,3469	2,5561
Iteracje:	6	15	Iteracje:	4	4
Przedział:	[-5 ; 1,4224]	[1,4224 ; 3,0264]	Przedział:	[-5 ; 1,4224]	[2,2224 ; 3,0264]
Wartości na krańcach:	[-24,089 ; 2,9647]	[2,9647 ; -5,8032]	Wartości na krańcach:	[-24,089 ; 2,9647]	[3,4295 ; -5,8032]

Wnioski

Obie metody prawidłowo policzyły miejsca zerowe. Możemy zauważyć, że metoda Newtona wykonała potrzebne obliczenia w mniejszej ilości iteracji. Wynika to z faktu, że funkcja jest bardzo stroma w okolicy miejsca zerowego, co jest pożądane dla metody, gdyż jej zbieżność jest kwadratowa, a więc większa od metody regula falsi, która jest zbieżna liniowo.

Metody Newtona nie zaleca się gdy funkcja w pobliżu pierwiastka jest prawie pozioma, bądź pochodna ma małą wartość.

Metoda regula falsi może być wolno zbieżna. Gdy jeden z końców izolacji pierwiastka pozostaje stały, metoda nie prowadzi do zmniejszenia do zera przedziału izolacji pierwiastka. Wymagana jest wtedy modyfikacja standardowego wzoru.

2.Używając metody M*ü*llera MM1, proszę znaleźć wszystkie pierwiastki wielomianu czwartego stopnia

$$f(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 \quad [a_4\ a_3\ a_1\ a_0] = [-2\ 3\ 6\ 2\ 3]$$

Metoda M*ü*llera

Metoda polega na aproksymacji wielomianu w otoczeniu rozwiązania funkcją kwadratową. Może być traktowana jako uogólnienie metody siecznych – zamiast interpolacji w dwóch punktach funkcją liniową (tzn. sieczną) wykonujemy interpolację w trzech punktach funkcją kwadratową. Istnieje również efektywna realizacja oparta na wykorzystaniu informacji o wielomianie jedynie w jednym punkcie, tzn. wykorzystująca do wyznaczenia funkcji kwadratowej wartości wielomianu i jego pierwszej i drugiej pochodnej w aktualnym punkcie. Występują dwie realizacje metody Müllera – MM1 oraz MM2. W zadaniu skupimy się na MM1.

Rozważmy trzy punkty x_0, x_1, x_2 wraz z wartościami wielomianu w tych punktach f(x_0), f(x_1) i f(x_2). Skonstruujemy funkcję kwadratową przechodzącą przez te punkty, a następnie wyznaczymy pierwiastki tej funkcji i potraktujemy jeden z nich jako kolejne, poprawione przybliżenie rozwiązania (pierwiastka wielomianu). Przyjmijmy, bez utraty ogólności, że x_2 jest aktualną aproksymacją rozwiązania (pierwiastka wielomianu). Wprowadzimy zmienną przyrostową

$$z=x-x_2$$

i wykorzystamy różnice

$$z_0 = x_0 - x_2$$

$$z_1 = x_1 - x_2$$

Oznaczając poszukiwaną parabolę przez

$$y(z) = az^2 + bz + c$$

Biorąc pod uwagę trzy dane punkty, mamy

$$az_0^2 + bz_0 + c = y(z_0) = f(x_0)$$

$$az_1^2 + bz_1 + c = y(z_1) = f(x_1)$$

$$c= y(0) = f(x_2)$$

Ponieważ interesuje nas pierwiastek paraboli o najmniejszym module (tzn. położony jak najbliżej x_2), więc do numerycznego wyznaczenia tego pierwiastka najlepiej wykorzystać wzory

$$z_+ = \frac{-2c}{b + \sqrt{b^2 - 4ac}}$$

$$z_- = \frac{-2c}{b - \sqrt{b^2 - 4ac}}$$

Do kolejnego przybliżenia rozwiązania bierzemy pierwiastek położony jak najbliżej x_2, tj. o mniejszym module

$$x_3 = x_2 + z_{min}$$

Przed przejściem do następnej iteracji odrzucamy spośród x_0, x_1, x_2 punkt położony najdalej od ostatnio wyznaczonego przybliżenia rozwiązania, tj. punktu x_3.

Definicje i wzory na podstawie książki Metody numeryczne prof. P. Tatjewskiego; OWPW, Warszawa 2013, wykresy do zad 1 z Wikipedii

Implementacja Metody M*ü*llera MM1 w Matlab

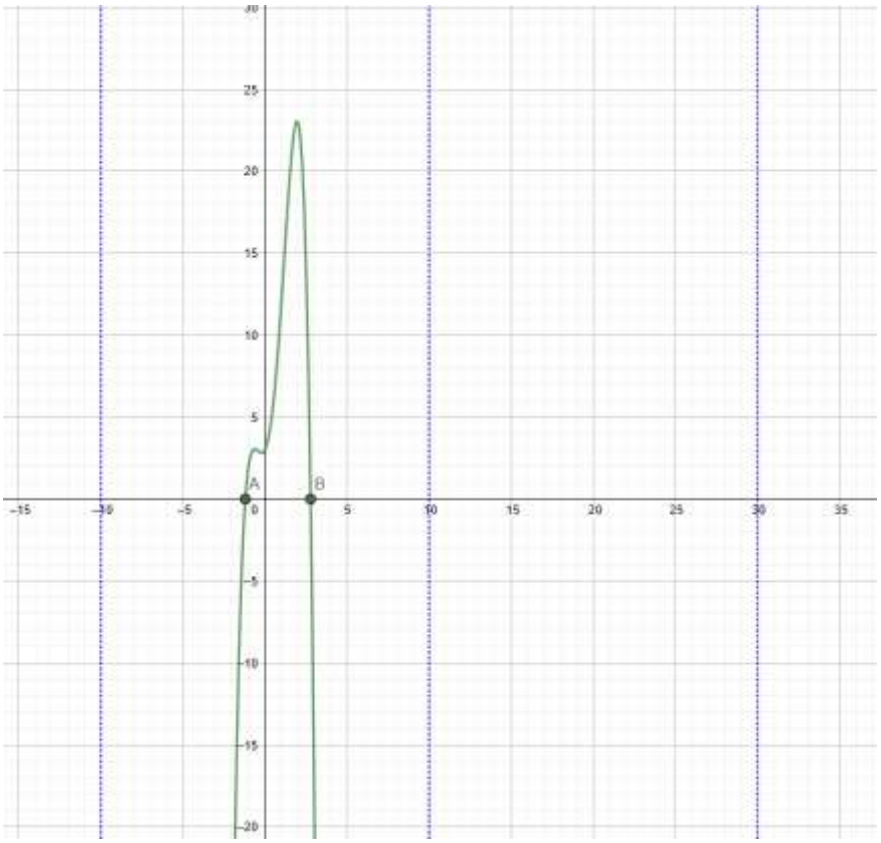
```
function [X3,n,F3] = mullermm1_solver(f,dokl,x0,x1,x2,n)

f0 = f(x0);
f1 = f(x1);
f2 = f(x2);
iteracje=0;
for i=1:n
```

```
    if i>n
        break
    end
    z01=x0-x1;
    z02=x0-x2;
    z12=x1-x2;
    y02=f0-f2;
    y12=f1-f2;
    %wspolczynniki paraboli
    a= (y02/z02 - y12/z12)/z01;
    b= y12/z12 - a*(x1 + x2);
    c= f2 - x2 * (a * x2 + b);
    d= b * b - 4 * a * c;
    d=sqrt(d);
    %pierwiastki
    zplus=(-2*c)/(b+d);
    zminus=(-2*c)/(b-d);

    if(abs(zplus-x2)<abs(zminus-x2))
        X3=zplus;
    else
        X3=zminus;
    end
    F3=f(X3);
    if(abs(F3)<dokl)||(abs(X3-x2)<dokl)
        break
    end
    x0=x1;
    x1=x2;
    x2=X3;
    f0=f1;
    f1=f2;
    f2=F3;
    iteracje=iteracje+1;
end
```

Funkcja wywołująca solver

```
f=@(x) -2*(x^4) +3*(x^3)+6*(x^2)+2*(x)+3;
dokl=10^-6;
n=70;
x0=-10;
x1=10;
x2=30;

[zerowe1,iteracje1,wartosc1]=mullermm1_solver(f,dokl,x0,x1,x2,n);
%stosuję deflacje czynnikiem liniowym
f2=@(x)f(x)./(x-zerowe1);
[zerowe2,iteracje2,wartosc2]=mullermm1_solver(f2,dokl,x0,x1,x2,n);
%stosuję deflacje czynnikiem liniowym
f3=@(x)f2(x)./(x-zerowe2);
[zerowe3,iteracje3,wartosc3]=mullermm1_solver(f3,dokl,x0,x1,x2,n);
%stosuję deflacje czynnikiem liniowym
f4=@(x)f3(x)./(x-zerowe3);
[zerowe4,iteracje4,wartosc4]=mullermm1_solver(f4,dokl,x0,x1,x2,n);
```

Wykres



Wyniki

		Metoda M*ü*llera MM1		
Wyniki :	-0.0427 + 0.6711i	-1,1936	2,7789	-0,0427 – 0,6711i
Iteracje:	70	70	70	70

Punkty początkowe:	-10	10	30
Wartości w punktach:	-22417	-16377	-1533537

Wnioski

Metoda dokładnie policzyła miejsca zerowe wielomianu, zostały policzone z sporą dokładnością. Metoda M*ü*llera ma dużą przewagę nad metodami z zdania pierwszego poprzez możliwość poruszania się po dziedzinie zespolonej i obliczania rozwiązań zespolonych.