Jan Jędrzejewski 310 713
22.05.2023r. - grupa 103

MNUM - PROJEKT 2

zadanie 2.29

Zadanie 1

Proszę znaleźć wszystkie pierwiastki funkcji

$$f(x) = -1.65 + 0.3 * x - x * e^{-x/2}$$

a) używając własnego solwera z implementacją metody siecznych
b) podanego na stronie przedmiotu solwera newton.m

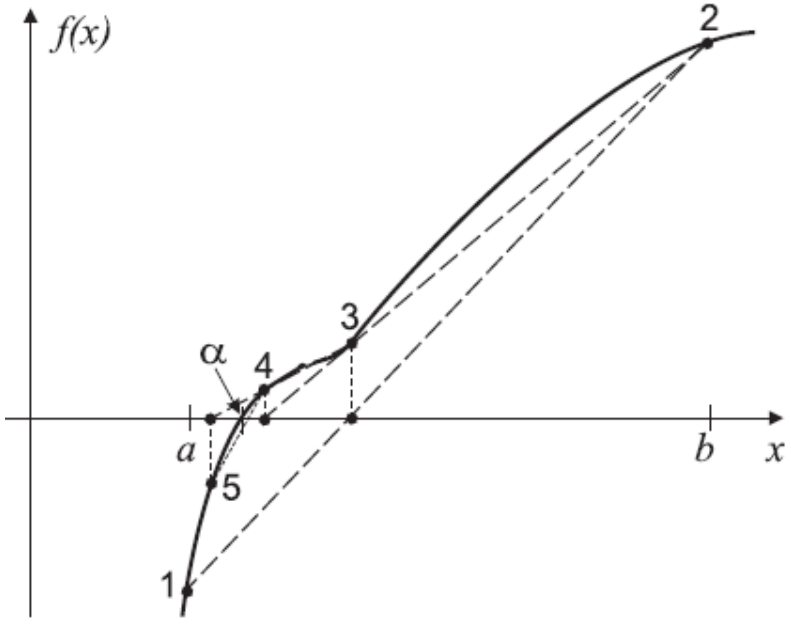
Zadana funkcja

```
function y = f(x)
   y = -1.65 + 0.3*x - x*exp(-x/2);
end
```

Metoda Siecznych

W metodzie siecznych prowadzimy sieczą zawsze między dwoma ostatnio wyznaczonymi punktami. W celu wyznaczenia kolejnego punktu wyznaczamy miejsce przecięcia siecznej z prostą y=0, kolejny punkt to współrzędna x tego punktu.

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



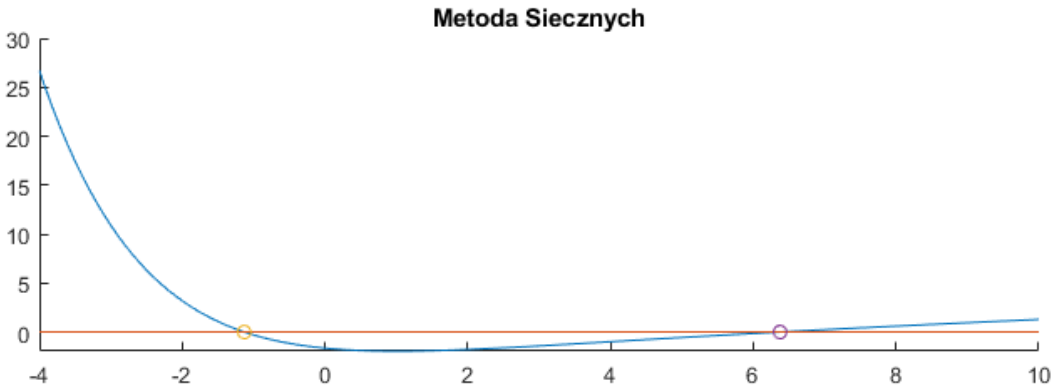
Implementacja

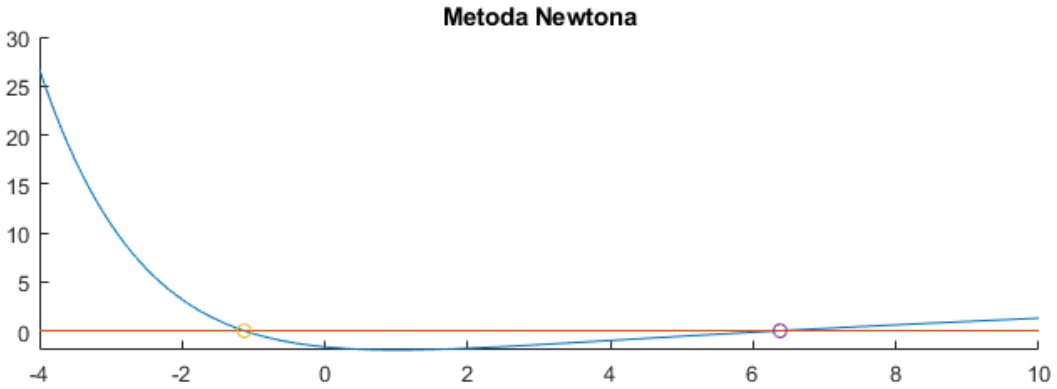
```
function [xf, ff, iexe, texe] = sieczne(f, x0, x1, delta, imax)
%   CEL
%       Poszukiwanie pierwiastka funkcji jednej zmiennej
%       metoda Newtona (stycznych)
%
%   PARAMETRY WEJSCIOWE
%       f      -  funkcja dana jako wyrazenie
%       x0, x1 -  przedział początkowy
%       delta  -  dokladnosc
%       imax   -  maksymalna liczba iteracji
%
%   PARAMETRY WYJSCIOWE
%       xf     -  rozwiazanie
%       ff     -  wartosc funkcji w xf
%       iexe   -  liczba iteracji wykonanych
%       texe   -  czas obliczen [s]
tic;
i = 0;
x_n_1 = x0;
x_n = x1;
fx = f(x_n);
while abs(fx) > delta && i < imax
    i = i + 1;
    x_tmp = (x_n_1 * f(x_n) - x_n * f(x_n_1)) / (f(x_n) - f(x_n_1));
    x_n_1 = x_n;
    x_n = x_tmp;
    fx = f(x_n);
end
texe = toc;
iexe = i;
xf = x_n;
ff = fx;
```

Pierwiastki funkcji

metoda	**x**	**f(x) [1e-08]**	**Iteracje**	**Czas**
siecznych	-1.1303	-0.0000	9	0.0016
siecznych	6.3766	0.1297	5	0.0003
Newtona	-1.1303	0.7618	6	0.0003
Newtona	6.3766	-0.0001	4	0.0003

Wykresy





Zestawienie wyników

Metoda siecznych

[pp,pk]	[f(pp),f(pk)	x	f(x)	Iter	Time
[-4.000000, -2.000000]	[26.706224, 3.186564]	-1.130336	0.0000000000	7	0.000042
[-2.000000, 0.000000]	[3.186564, -1.650000]	-1.130336	-0.0000000010	7	0.000034
[0.000000, 2.000000]	[-1.650000, -1.785759]	6.376608	0.0000000010	8	0.000039
[2.000000, 4.000000]	[-1.785759, -0.991341]	6.376608	0.0000000027	4	0.000027
[4.000000, 6.000000]	[-0.991341, -0.148722]	6.376608	-0.0000000000	4	0.000014
[6.000000, 8.000000]	[-0.148722, 0.603475]	6.376608	0.0000000000	4	0.000024
[8.000000, 10.000000]	[0.603475, 1.282621]	6.376608	0.0000000000	5	0.000018
[10.000000, 12.000000]	[1.282621, 1.920255]	6.376608	0.0000000000	5	0.000014
[12.000000, 14.000000]	[1.920255, 2.537234]	6.376608	0.0000000002	5	0.000020

Metoda Newtona

pp	f(pp)	x	f(x)	Iter	Time
-4.000000	26.706224	-1.130336	0.0000000076	6	0.000305
-2.000000	3.186564	-1.130336	0.0000000000	5	0.000009
0.000000	-1.650000	-1.130336	0.0000000000	6	0.000010
2.000000	-1.785759	6.376608	-0.0000000003	4	0.000009
4.000000	-0.991341	6.376608	-0.0000000012	3	0.000010
6.000000	-0.148722	6.376608	-0.0000000000	3	0.000009
8.000000	0.603475	6.376608	-0.0000000004	3	0.000009
10.000000	1.282621	6.376608	-0.0000000000	4	0.000018
12.000000	1.920255	6.376608	-0.0000000000	4	0.000009

Wnioski

Obie metody dobrze poradziły sobie ze znalezieniem miejsc zerowych funkcji w podobnej liczbie iteracji, w podobnym czasie. i z podobną dokładnością. W metodzie Newtona zaobserwowałem nieznacznie krótsze czasy wykonania i kilka mniej wykonanych iteracji. Wyraźnie widać przedziały izolacji pierwiastków, w obydwu algorytmach w okolicy 0. Dla pierwszego uruchomienia w tabeli widać wyraźnie wyższe czasy wykonania, spowodowane prawdopodobnie ładowaniem potrzebnych zasobów. Podsumowując, obie metody poprawnie działają i znajdują pierwiastki z podaną dokładnością.

Zadanie 2

Używając metody M¨ullera MM1 proszę znaleźć wszystkie pierwiastki wielomianu czwartego stopnia:

$$f(x) = -1 * x^4 + 1.5 * x^3 + 3 * x^2 + 1 * x + 1.5$$

Zadana funkcja

```
function y = f(x)
   y = -1*x^4 + 1.5*x^3 + 3*x^2 + 1*x + 1.5;
end
```

Metoda Mullera MM1

Rozważamy trzy punkty x_0. x_1, x_2 wraz z wartościami wielomianu w tych punktach. Przez te punkty prowadzimy parabolę, a kolejne przybliżenia pierwiastka wyznaczamy korzystając z wyliczonego pierwiastka paraboli. Przyjmujemy że x_2 jest aktualną aproksymacją rozwiązania. Wprowadźmy zmienną przyrostową z

$$z = x - x_2$$

$$z_0 = x_0 - x_2$$

$$z_1 = x_1 - x_2$$

Oznaczając poszukiwaną parabolę przez

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

Rozwiązujemy układ równań z 3 niewiadomymi a, b, c:

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

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

$$c = y(z_2) = f(x_2)$$

Z ostatniego równania bezpośrednio otrzymujemy c i podstawiamy, aby wyznaczyć a i b.

$$az_0^2 + bz_0 = f(x_0) - f(x_2)$$

$$az_1^2 + bz_1 = f(x_1) - f(x_2)$$

Układ z dwoma niewiadomymi rozwiązujemy w następujący sposób, z pierwszego równania wyznaczamy a i wstawiamy do drugiego, obliczamy b i podstawiamy żeby wyliczyć a.

$$a = \frac{f(x_0) - f(x_2) - bz_0}{z_0^2}$$

$$\frac{f(x_0) - f(x_2) - bz_0}{z_0^2} z_1^2 + bz_1 = f(x_1) - f(x_2)$$

$$(f(x_0) - f(x_2) - bz_0)z_1^2 + bz_1 z_0^2 = z_0^2[f(x_1) - f(x_2)]$$

$$f(x_0)z_1^2 - f(x_2)z_1^2 - bz_0 z_1^2 + bz_1 z_0^2 = z_0^2[f(x_1) - f(x_2)]$$

$$- bz_0 z_1^2 + bz_1 z_0^2 = z_0^2[f(x_1) - f(x_2)] - f(x_0)z_1^2 + f(x_2)z_1^2$$

$$b(z_1 z_0^2 - z_0 z_1^2) = z_0^2[f(x_1) - f(x_2)] - f(x_0)z_1^2 + f(x_2)z_1^2$$

$$b = \frac{z_0^2[f(x_1) - f(x_2)] + z_1^2[f(x_2) - f(x_0)]}{z_1 z_0^2 - z_0 z_1^2}$$

Następnie wyznaczamy pierwiastek paraboli położony jak najbliżej x_2, ze wzorów o najlepszym uwarunkowaniu numerycznym:

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

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

Kolejne przybliżenie rozwiązania to:

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

Spośród punktów x_0. x_1, x_2 odrzucamy punkt położony najdalej od ostatnio wyznaczonego przybliżenia rozwiązania tj. punktu x_3.

Implementacja

```
function [XF, FF, iexe] = MM1(A, x0, x1, x2, delta)
%       Poszukiwanie pierwiastków wielomianu
%       metoda Mullera MM1
%   PARAMETRY WEJSCIOWE
%       A           -  wektor współczynników wielomianu
%       x0, x1, x2  -  punkty początkowe
%       delta       -  dokładność
%   PARAMETRY WYJSCIOWE
%       Xf      -  wektor pierwiastków wielomianu
%       FF      -  wektor wartości pierwiastków
%       iexe    -  liczba iteracji wykonanych
   n = length(A);
   XF = zeros(1, n-1);
   FF = zeros(1, n-1);
   iexe = 0;
   X = [x0, x1, x2];
   A_i = A;
   for i = 1:n-1
       while abs(polyval(A_i, X(3))) > delta
           z0 = X(1) - X(3);
           z1 = X(2) - X(3);
           fx0 = polyval(A_i, X(1));
           fx1 = polyval(A_i, X(2));
           c = polyval(A_i, X(3));
           b = (z0*z0*(fx1 - c) + z1*z1*(c - fx0)) / (z1*z0*z0 - z0*z1*z1);
           a = (fx0 - c - b*z0) / (z0*z0);
           sqrt_delta = sqrt(b^2 - 4 * a * c);
           if (abs(b + sqrt_delta) >= abs(b - sqrt_delta))
               zmin = - 2 * c / (b + sqrt_delta);
           else
               zmin = - 2 * c / (b - sqrt_delta);
           end
           x3 = X(3) + zmin;
           diff_max = 0;
           val_max = 0;
           for j = 1:3
               if diff_max < abs(x3 - X(j))
                   diff_max = abs(x3 - X(j));
                   val_max = X(j);
               end
           end
           X = [X(X~=val_max), x3];
           iexe = iexe + 1;
       end
       A_i = deflate(X(3), A_i);
       XF(i) = X(3);
       FF(i) = polyval(A, X(3));
   end
end
```

Deflacja czynnikiem liniowym

Aby uniknąć wyznaczania ponownie tych samych zer wielomianu stosujemy deflację liniową, która polega na dzieleniu wielomianu przez wyznaczony pierwiastek. Dzięki temu zmniejszamy stopień wielomianu i wykluczamy już odkryte zera.

Schemat Hornera

Oznaczmy .$Q(x)$ - wielomian po dzieleniu przez $(x - \alpha)$ funkcji $f(x)$, gdzie α to pierwiastek funkcji. Wówczas:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ \dots \ + a_1 x^1 + a_0 = (x - \alpha) * Q(x)$$

Wyznaczmy $Q(x)$ korzystając ze schematu Hornera:

$$q_{n+1} = 0$$

$$q_i = a_i + q_{i+1} * \alpha$$

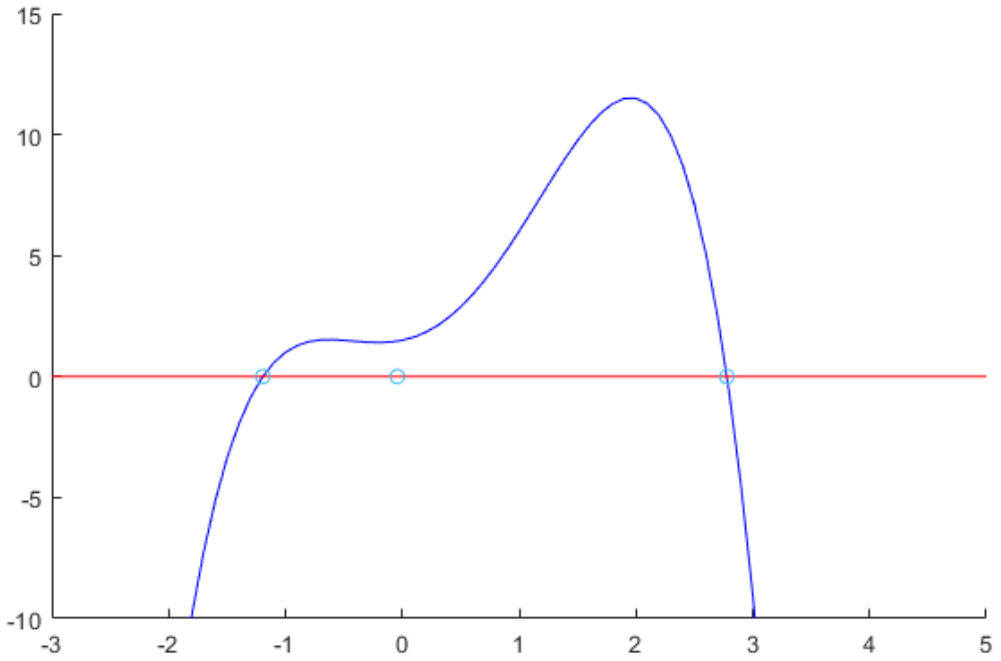
$$i = n, n-1, \dots, 1, 0$$

Implementacja

W poniższej implementacji wektor współczynników A jest ustawiony od najwyższego stopnia do najniższego. Zatem biorąc z powyższego wzoru q_n to w matlabie Q(1) a q_1 to Q(n)

```
function Q = deflate(alpha, A)
%   CEL
%       Deflacja czynnikiem liniowym - prosty Schemat Hornera
%
%   PARAMETRY WEJSCIOWE
%       alpha  -  pierwiastek wielomianu
%       A      -  wektor współczynników wielomianu
%
%   PARAMETRY WYJSCIOWE
%       Q      -  wektor współczynników wielomianu po deflacji
%
   n = length(A);
   Q = zeros(1, n-1);
   Q(1, 1) = A(1);
   for i = 2:n-1
       Q(1,i) = A(i) + Q(1, i-1) * alpha;
   end
end
```

Wykresy



x	**f(x)**	**Iteracje**
-0.042670+0.671122j	0.000000-0.000000j	8
-1.193597-0.000000j	0.000000-0.000000j	6
-0.042670-0.671122j	0.000000-0.000000j	2
2.778938+0.000000j	0.000000-0.000000j	1

Wnioski

Metoda Mullera MM1 poprawnie znalazła wszystkie pierwiastki wielomianu. Działa na liczbach zespolonych, skąd otrzymaliśmy dodatkowe dwa pierwiastki w dziedzinie zespolonej. Liczba iteracji maleje z każdym pierwiastkiem, ponieważ w wyniku deflacji z każdą iteracją szukamy pierwiastka wielomianu o niższym stopniu.