Tomasz Indeka

Metody numeryczne

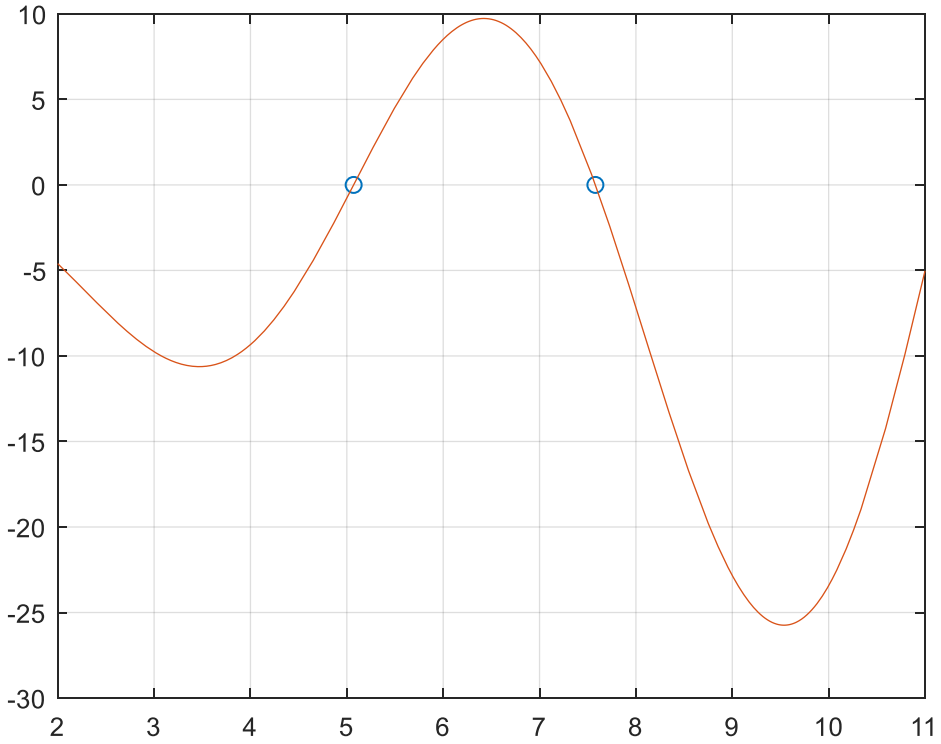
Zadanie 3.15 – Sprawozdanie

Pkt 1. Poszukiwanie zer funkcji w przedziale

Funkcja w której poszukiwane są pierwiastki:

$f(x) = 2.2x\cos(x) - 2\ln(x+2)$, w przedziale $< 2, 11 >$

Wykres funkcji f(x):



Metoda regula falsi polega na podzieleniu aktualnego przedziału na 2 przez poprowadzenie prostej siecznej i następnie zawężenie przedziału przez wyznaczenie iloczynów funkcji w tym puncie i na krańcach przedziału i wyborze przedziału zawierającego pierwiastek. Później zmniejszamy przedział aż do zadowalającej dokładności. Metoda może zawieść kiedy jeden z punktów pozostaje niezmienny przez dłuższy czas, a drugi jest blisko zera. Wtedy metoda jest wolno zbieżna lub nawet podaje błędny wynik.

Do wyznaczenia wartości pierwiastka użyłem wzoru:

$$c = \frac{a * f(b) - b * f(a)}{f(b) - f(a)}$$

Metoda siecznych różni się od metody regula falsi tym że sieczną zawsze prowadzimy między dwoma ostatnio wyznaczonymi punktami, nie dbając o zachowanie przedziału izolacji pierwiastka. Dokonujemy obliczeń aż do zadowalającej dokładności. Metoda może zawieść kiedy początkowy przedział izolacji pierwiastka jest zbyt duży. Np dla podania danych przedziału izolacji pierwiastka

(6,2) (przedział odwrócony, aby zmienić kolejnością punkty 1 i 2) i dokładności 0.001, algorytm znajduje pierwiastek po 16 krokach nienależący do przedziału izolacji pierwiastka: -1.3101 - 0.0002i.

Do wyznaczenia wartości pierwiastka użyłem wzoru (bardzo podobnego do powyższego):

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

Metoda Newtona (stycznych) polega na aproksymacji funkcji jej liniowym przybliżeniem w postaci stycznej w punkcie i zawężaniu przedziału od jednej strony. Dokonujemy obliczeń aż do zadowalającej dokładności. Metoda może zawieść kiedy pochodna w okolicach pierwiastka jest bardzo mała (krzywa jest prawie pozioma).

Do wyznaczenia wartości pierwiastka użyłem wzoru:

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

Wartości przybliżenia pierwiastka w kolejnych iteracjach przy dokładności 0.001

Przedział	5-6					
Metoda	Regula falsi		Sieczne		Newton	
Numer iteracji	x	f(x)	x	f(x)	x	f(x)
1	5.0831	0.1356	5.0831	0.1356	5.0709	0.0021
2	5.0682	-0.0267	5.0682	-0.0267	**5.0706**	**-0.0004**
3	5.0711	0.0051	**5.0707**	**-0.0000**		
4	**5.0706**	**-0.0010**				

Przedział	7-8					
Metoda	Regula falsi		Sieczne		Newton	
Numer iteracji	x	f(x)	x	f(x)	x	f(x)
1	7.5017	1.1911	7.5017	1.1911	7.8312	-4.1783
2	7.5727	0.1061	7.5727	0.1061	7.5951	-0.2447
3	7.5790	0.0087	7.5797	-0.0025	7.5808	-0.0194
4	**7.5795**	**0.0007**	**7.5795**	**0.0000**	7.5796	-0.0016
5					**7.5795**	**-0.0001**

Metoda regula falsi:

```
%f-funkcja
%n-ilość pierwiastków
%p-początek przedziału
%z-koniec przedziału
%tol-tolerancja dokładności pierwiatków
%a-początek przedziału sprawdzanego
%k,b-koniec przedziału sprawdzanego
```

```
%c-przybliżony pierwiastek
%y-wszytkie pierwiastki funkcji na przedziale
%h-ilosc iteracji przyblizania pierwiastka
%g-kolejne przyblizone pierwiastki
%j-kolejne wartości przybliżonego pierwiastka

function [y,g,j] = falsi(p,z,tol)
f=@(x)(2.2*x*cos(x)-2*log(x+2)); % funkcja
z=z+1;
k=p+1;
n=0;
while k<z
    if f(p)*f(k)<0 % sprawdzanie wszystkich przedziałów (co 1) czy istnieje
pierwiastek
        a=p;
        b=k;
        n=n+1;
        h=1;
c=(a*f(b)-b*f(a))/(f(b)-f(a)); % poszukiwanie pierwiastka
g(h)=c;
j(h)=f(c);
while abs(f(c))>tol
    if a*c<0
        b=c;
    else
        a=c;
    end
    c=(a*f(b)-b*f(a))/(f(b)-f(a));
    h=h+1;
    g(h)=c;
    j(h)=f(c);
end
y(n)=c;
    end
p=p+1;
k=k+1;
end
end
```

Metoda siecznych:

```
%f-funkcja
%n-ilość pierwiastków
%p-początek przedziału
%z-koniec przedziału
%tol-tolerancja dokładności pierwiatków
%a-początek przedziału sprawdzanego
%k,b-koniec przedziału sprawdzanego
%c-przybliżony pierwiastek
%y-wszytkie pierwiastki funkcji na przedziale
%h-ilosc iteracji przyblizania pierwiastka
%g-kolejne przyblizone pierwiastki
%j-kolejne wartości przybliżonego pierwiastka

function [y,g,j] = sieczne (p,z,tol)
f=@(x)(2.2*x*cos(x)-2*log(x+2)); % funkcja
z=z+1;
k=p+1;
n=0;
```

```
while k<z
    if f(p)*f(k)<0 % sprawdzanie wszystkich przedziałów (co 1) czy istnieje
pierwiastek
        a=p;
        b=k;
        n=n+1;
        h=1;
c=(a*f(b)-b*f(a))/(f(b)-f(a)); % poszukiwanie pierwiastka
g(h)=c;
j(h)=f(c);
while abs(f(c))>tol
    a=b;
    b=c;
    c=(a*f(b)-b*f(a))/(f(b)-f(a));
    h=h+1;
    g(h)=c;
    j(h)=f(c);
end
y(n)=c;
    end
p=p+1;
k=k+1;
end
end
```

Metoda Newtona:

```
%f-funkcja
%df-pochodna funkcji
%n-ilość pierwiastków
%p-początek przedziału
%z-koniec przedziału
%tol-tolerancja dokładności pierwiatków
%a-początek przedziału sprawdzanego
%k,b-koniec przedziału sprawdzanego
%c-przybliżony pierwiastek
%y-wszytkie pierwiastki funkcji na przedziale
%h-ilosc iteracji przyblizania pierwiastka
%g-kolejne przyblizone pierwiastki
%j-kolejne wartości przybliżonego pierwiastka

function [y,g,j] = newton(p,z,tol)
syms f(x) df(x);
f(x)=(2.2*x*cos(x)-2*log(x+2)); % funkcja
df(x) = diff(f(x)); % pochodna funkcji
z=z+1;
k=p+1;
n=0;
while k<z
    if f(p)*f(k)<0 % sprawdzanie wszystkich przedziałów (co 1) czy istnieje
pierwiastek
        a=p;
        b=k;
        n=n+1;
        h=1;
c=a-f(a)/df(a); % poszukiwanie pierwiastka
g(h)=c;
j(h)=f(c);
while abs(f(c))>tol
```

```
    a=c;
    c=(a*f(b)-b*f(a))/(f(b)-f(a));
    h=h+1;
    g(h)=c;
    j(h)=f(c);
end
y(n)=c;
    end
p=p+1;
k=k+1;
end
y=double(y);
g=double(g);
j=double(j);
end
```

Pkt 2. Poszukiwanie wszystkich zer funkcji wielomianowej za pomocą metody Mullera, polegającej na aproksymacji kwadratowej trzech punktów.

Metoda ta polega na aproksymacji kwadratowej trzech punktów i poszukiwania pierwiastka w tym zakresie. Później obliczony pierwiastek zastępuje najdalszy z punktów aproksymowanych i metoda jest powtarzana do momentu otrzymania pierwiastka wielomianu z zadowalającą dokładnością.

Do wyznaczenia współczynników a, b i c wielomianu aproksymującego użyłem wzorów:

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

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

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

Które po przekształceniu sprowadziły się do:

$$c = f(x_2)$$

$$a = \frac{\frac{f(x_1) - f(x_2)}{z_1} \_ \frac{f(x_0) - f(x_2)}{z_0}}{z_1 - z_0}$$

$$b = \frac{f(x_0) - f(x_2) - az_0^2}{z_0}$$

Później wybrałem mniejszy pierwiastek kwadratowy, aby szybciej znaleźć pierwiastek całego wielomianu i na jego podstawie przybliżałem pierwiastek wielomianu:

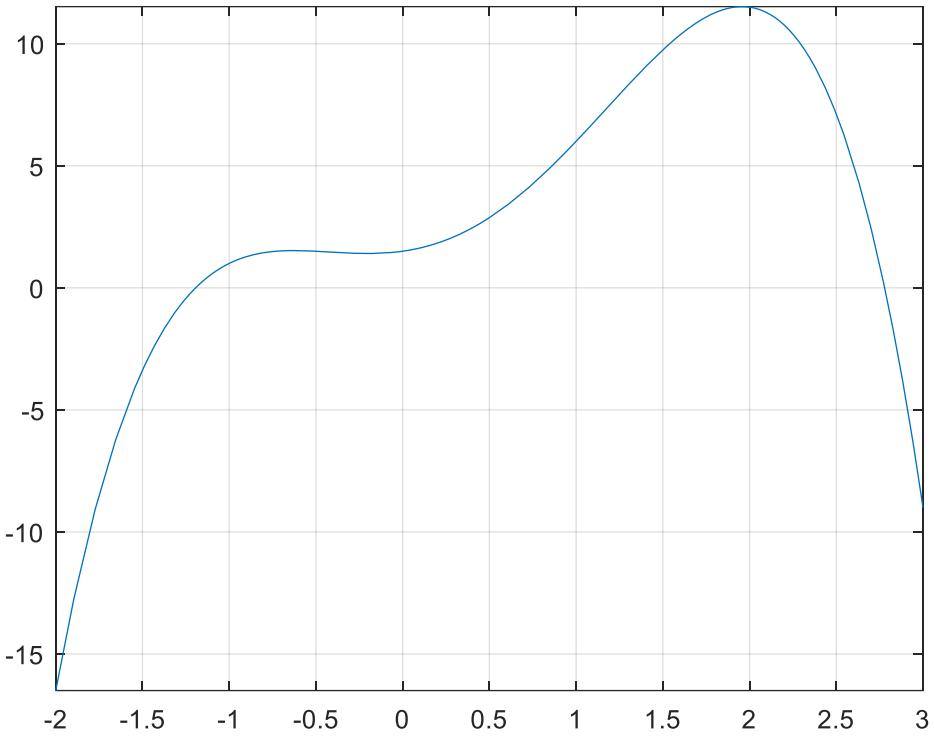
$$z_{min} = \min\left(\frac{-2c}{b \pm \sqrt{b^2 - 4ac}}\right)$$

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

Wielomian podany w zadaniu:

$-x^4 + 1.5x^3 + 3x^2 + x + 1.5$

Wykres wielomianu w okolicach 0:



Obliczone pierwiastki wielomianu z dokładnością do 0.000001:

-1,19359 - 6,67473e-06i

-0,04267 + 0,67112i

-0,04266 - 0,67111i

2,77893 + 1,26590e-06i

Kod programu zawierający dane:

```
%a-wektor współczynników wielomianów przy kolejnych potęgach x
%tol-tolerancja dokładności pierwiatków
%y-wszytkie pierwiastki funkcji wielomianowej

a=[-1 1.5 3 1 1.5];
tol=0.000001;
y=muller(a,tol)
```

Metoda MM1:

```
%w-funkcja
%n-ilość pierwiastków
```

```
%tol-tolerancja dokładności pierwiatków
%y-wszytkie pierwiastki funkcji na przedziale
%x-punkty do aproksymacji i szukania pierwiastka
%z,zc,zd-przewidywany pierwiastek

function y = muller(w,tol)
n = size(w)-1;
n=n(2);
while n>0 % szukanie n pierwiastków wielomianu w
zd=1;
x=[1 2 3];
while zd>tol % poszukiwanie pojedynczego pierwiastka
z=x-x(3);
c=polyval(w,x(3));
a=((polyval(w,x(2))-polyval(w,x(3)))/z(2)-(polyval(w,x(1))-
polyval(w,x(3)))/z(1)/(z(2)-z(1)));
b=(polyval(w,x(1))-polyval(w,x(3))-a*z(1)*z(1))/z(1);
za=(-2*c)/(b+sqrt(b^2-4*a*c));
zb=(-2*c)/(b-sqrt(b^2-4*a*c));
zd=min(abs(za),abs(zb)); % wybór przybliżenia pierwiastka bliżej zera
if abs(za)>abs(zb)
zc=x(3)+zb;
else
zc=x(3)+za;
end
if abs(x(1)-zc)>abs(x(2)-zc) && abs(x(1)-zc)>abs(x(3)-zc) % eliminacja
pierwiastka najbardziej oddalonego od aktualnie obliczonego
x(1)=x(3);
x(3)=zc;
elseif abs(x(2)-zc)>abs(x(3)-zc)
x(2)=x(3);
x(3)=zc;
else
x(3)=zc;
end
end
y(n)=zc; % wypisanie pierwiastka
[w,r]=deconv(w,[1 -zc]); %deflacja czynnikiem liniowym
n=n-1;
end
end
```