Porównanie zbieżności wybranych metod iteracyjnych rozwiązywania równań nieliniowych

Projekt zespołowy

Beata Gajewska, Agnieszka Wrzos, Artur Moczybroda

8 kwietnia 2019

1 Wstęp

Metody numeryczne zajmują się badaniem sposobów umożliwiających rozwiązywanie zadań matematycznych za pomocą działań arytmetycznych. Nierzadko podczas wyznaczania miejsc zerowych programy stosują różnego rodzaju przybliżenia. W takich przypadkach mamy do czynienia z błędami związanymi z aproksymacją funkcji. Celem naszego projektu jest analiza sposobów wyznaczania miejsc zerowych w programie **Octave**. Posłużymy się utworzonymi przez nas skryptami bazującymi na metodach: Newtona, bisekcji i siecznych. Ponadto porównamy je z funkcjami wbudowanymi typu fsolve i fzero. Wykonamy eksperymenty numeryczne na obranych przez nas funkcjach, by zbadać, która metoda jest najbardziej wydajna i dokładna. Zestawimy wyniki w tabelach w celu ułatwienia analizy zebranych danych.

2 Skrypt głównego programu

```
%program główny
format long
printf('Witamy w programie do szukania miejsca zerowego funkcji. \n');
printf('Wybierz funkcję poprzez wpisanie odpowiedniej cyfry: \n');
printf('1 - f(x)=log10(x)-1 \n');
printf('2 - f(x)=x-x^(1/3)-0.01 \n');
printf('3 - f(x)=sin(x-3)x^2 \n');
printf('4 - f(x)=a*x^5+b*x^4+c*x^3+d*x^2+e*x+f \n');
n=input('Numer funkcji: '); % podanie cyfry
if(n==1) % sprawdzenie jaka podana została cyfra i przypisanie odpowiedniej
f=@(x) log10(x)-1;   %funkcji do programu
elseif(n==2)
f=@(x) x-x.^(1/3)-0.01;
elseif(n==3)
f=@(x) sin(x-3)*x.^2;
elseif(n==4)
printf('Podaj współczynniki wielomianu \n');
a1=input('a= ');
a2=input('b= ');
a3=input('c= ');
a4=input('d= ');
a5=input('e= ');
a6=input('f= ');
f=@(x) a1*x.^5+a2*x.^4+a3*x.^3+a4*x.^2+a5*x+a6;
else
printf('Podano złą cyfrę \n')
endif
printf('Wybierz metodę poprzez wpisanie odpowiedniej cyfry: \n');
printf('1 - bisekcja \n');
```

```
printf('2 - sieczna \n');
printf('3 - newton \n');
m=input('Numer metody: '); % podanie cyfry
if(m==1) %sprawdzenie wpisanej cyfry i uruchomienie odpowiedniej metody
bisekcja(f);
elseif(m==2)
sieczna(f);
elseif(m==3)
newton(f);
else
printf('Podano złą cyfrę \n')
endif
```

3 Skrypt programu pomocniczego - obliczanie pochodnej w punkcie

W skrypcie pomocniczym zastosowaliśmy przybliżoną wartość oblliczania pochodnej w punkcie, której wartość ilorazu różnicowego dla bardzo małego przyrostu jest bardzo zbliżona do granicy z ilorazu różnicowego.

$$\lim_{t\to 0}\frac{f(x+t)-f(x)}{t} \approx \frac{f(x+t)-f(x)}{t}, t = 0.000001$$

```
function y=pochodna(f,x)
y=(f(x+0.000001)-f(x))/0.000001;
endfunction
```

4 Metoda bisekcji

Pod uwagę bierzemy przedziały liczbowe $[a, b]$, w których funkcja jest ciągła, a do tego spełniony jest poniższy warunek:

$$f(a) \cdot f(b) < 0$$

Zgodnie z twierdzeniem Darboux każda funkcja spełniająca powyższe warunki, ma na danym przedziale co najmniej jedno miejsce zerowe.

Sposób działania metody bisekcji jest następujący:
-szukamy takiego przedziału, aby wartości funkcji na jego końcach były różnych znaków,
-wyznaczamy średnią artmetyczną z wartości na końcach obranego przedziału,
-badamy wartość funkcji dla otrzymanej średniej arytmetycznej,
-jeśli jest ona dodatnia to do następnego kroku użyjemy jej oraz ujemnego końca przedziału, jeśli ujemna - dodatniego. Powtarzamy cały algorytm, aż do momentu otrzymania pożądanej dokładności.

4.1 skrypt

```
function bisekcja(f)
printf('Podaj: \n');
Eps=input('Błąd: '); %błąd z jakim szukamy miejsca zerowego
a=input('Początek przedziału: ');      %początek przedziału
b=input('Koniec przedziału: ');          %koniec przedziału
fa=f(a); %wartość początka przedziału
fb=f(b); %wartość końca przedziału
while(sign (fa) == sign (fb)) %porównanie znaków wartości funkcji na końcach
printf ('Zły przedział, końce przedziałów muszą mieć różne znaki \n');
a=input('Nowy początek przedziału: ');      %początek przedziału
b=input('Nowy koniec przedziału: ');          %koniec przedziału
fa=f(a); %wartość początku przedziału
fb=f(b); %wartość końca przedziału
endwhile
i=0; % i to ilość wykonanych obrotów pętli
while(abs(a-b)>2*Eps) % sprawdzenie długości przedziału czy większy od 2*Eps
s=(a+b)/2; % wyliczenie środka przedziału
fs=f(s); %wartość dla środka przedziału
i=i+1; % dodanie 1 do obrotu pętli
if (sign (fa) == sign (fs)) %porównanie znaków wartości funkcji w a i s
a=s; % jeśli wartości na początku i środku przedziału jest takie samo to
fa=fs; % zamiana środka przedziału na nowy początek
else  % analogicznie przypisanie wartości funkcji dla środka i początku
b=s; % jeśli znaki warości w początku i środku przedziału są różne
fb=fs; % to zamiana środka przedziału na nowy koniec
endif
endwhile
pierw=(a+b)/2; % miejscem zerowym jest środek wyliczonego przedziału
printf('Miejsce zerowe funkcji to: %.10f \n',pierw); %wyświetlenie wyników
printf('Wartość funkcji w wyznaczonym miejscu zerowym to: %.10f \n',f(pierw));
printf('Ilość wykonanych obrotów pętli: %d \n',i);
endfunction
```

4.2 fzero

Wbudowana funkcja *fzero()* wykorzystuje metodę bisekcji. Struktura funkcji *fzero()*:

- x0=fzero("funkcja",[tu podajemy przedział startowy])
- $[r, fr, zb, info]$ =fsolve("funkcja",[tu podajemy przedział startowy]) Gdzie:
  r - przybliżone miejsce zerowe
  fr - wartość funkcji w punkcie r zb - informacja o wyniku wykonania,

rodzaj zbieżności (wartość 1 oznacza sukces),
$info$ - komunikat dla użytkownika.

5 Metoda Newtona

Metoda ta, zwana również metodą stycznych geometrycznie polega na tzw. linearyzacji. Krzywą $y = f(x)$ przybliżamy w odpowiednim punkcie odcinkiem stycznej do tej krzywej.
W przedziale poszukiwań pierwiastka funkcja musi spełniać następujące warunki:

- funkcja $f(x)$ jest określona
- funkcja $f(x)$ jest ciagła
- funkcja $f(x) \in C^2$

Sposób działania metody Newtona jest następujący:
-wyznaczamy punkt startowy,
-konstruujemy ciąg przybliżeń rozwiązania według zasady, że każde kolejne przybliżenie jest równe różnicy poprzedniego przybliżenia i ilorazowi funkcji w punkcie x_0 oraz pochodnej tej funkjci również w punkcie x_0,
- powtarzamy cały algorytm, aż do momentu otrzymania pożądanej dokładności.

5.1 skrypt

```
function newton(f)
printf('Podaj: \n');
Eps=input('Błąd: '); %błąd z jakim szukamy miejsce zerowe
a=input('Punkt początowy '); %punkt początkowy
c(1)=a; % tworzymy wektor i definiujemy punkt początkowy jako pierwsza
%współrzędna
c(2)=c(1)-f(c(1))/pochodna(f,c(1)); %obliczenie drugiej współrzędnej wektora
% jako miejsce przecięcia stycznej z punktu początkowego i osi y=0
i=0; % i to ilość obrotów pętli
k=2; % k to numer współrzędnej wektora
while(abs(c(k)-c(k-1))>Eps) % sprawdzenie długości przedziału czy większy o
%epsilon czyli przyjęty błąd
c(k+1)=c(k)-f(c(k))/pochodna(f,c(k)); %obliczenie kolejnego punktu jako
% przecięcie stycznej z punktu poprzedniego i osi y=0
k=k+1; % dodanie jednej współrzędnej do wektora
i=i+1; % dodanie jednego obrotu do pętli
endwhile
printf('Miejsce zerowe funkcji to: %.10f \n',c(k)) % wyświetlenie wyników
printf('Wartość funkcji w wyznaczonym miejscu zerowym to: %.10f \n',f(c(k)))
```

```
printf('Ilość wykonanych obrotów pętli: %d \n',i)
endfunction
```

5.2 fsolve

Wbudowana funkcja *fsolve* implementuje metodę podobną do metody Newtona. Składnia funkcji *fsolve* jest następująca:

- x0=fsolve("funkcja",tu podajemy punkt startowy)
- fsolve_options("tolerance",tu podejamu jakiej dokładności chcemy przykład: 1e-5);
  fsolve("funkcja",tu podajemy punkt startow);
  (funkcja pozwala również na ręczne ustalenie dopuszczalnego błędu)
- $[x, info, msg]$=fsolve("funkcja",tu podajemy punkt startowy)
  Gdzie:
  x - przybliżone miejsce zerowe
  $info$ - informacja o wyniku wykonania (wartość 1 oznacza sukces),
  msg - komunikat dla użytkownika.

6 Metoda siecznych

Podobnie jak w metodzie bisekcji należy podać przedział $[a, b]$, dla którego:

- funkcja $f(x)$ jest określona
- funkcja $f(x)$ jest ciągła
- $f(a) \cdot f(b) < 0$

Ponadto:

- dla wybranego punktu x_0 jest, że $f'(x_0) \neq 0$. Nie istnieje zatem minimum lub maksimum lokalne. Ten warunek gwarantuje nam, iż styczna nie będzie równoległa do osi OX, co uniemożliwiłoby wyznaczenie jej punktu przecięcia z tą osią.

Gdy funkcja $f(x)$ spełnia powyższe warunki, to w przedziale $[a, b]$ zagwarantowane jest istnienie pierwiastka i możemy go wyszukać przy użyciu metody siecznych.
Sposób działania metody siecznych jest następujący:
-szukamy takiego przedziału, aby wartości funkcji na jego końcach były różnych znaków,
-konstruujemy ciąg przybliżeń rozwiązań, który dodatkowo musi być nierosnący. Szukamy miejsca zerowego według zasady, że każde kolejne przybliżenie jest równe różnicy poprzedniego przybliżenia i ilorarowi funkcji w punkcie poprzedniego przybliżenia z ilorazem różnicy $x_n - x_{n-1}$ oraz różnicy wartości funkjci w

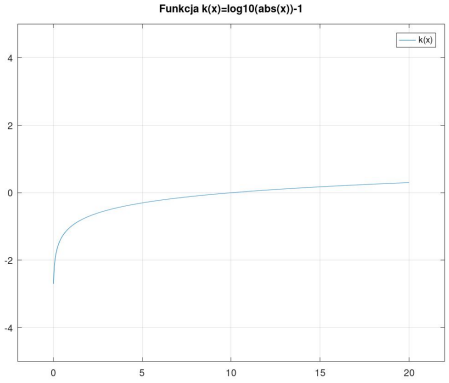
punkcie x_n i wartości funkcji w punkcie x_{n-1} (co tutaj pełni funkcję przybliżonej pochodnej) ,
- powtarzamy cały algorytm, aż do momentu otrzymania pożądanej dokładności.

6.1 skrypt

```
function sieczna(f)
printf('Podaj: \n');
Eps=input('Błąd: '); %błąd z jakim szukamy miejsca zerowego
a=input('Początek przedziału: ');      %początek przedziału
b=input('Koniec przedziału: ');          %koniec przedziału
fa=f(a); %wartość początka przedziału
fb=f(b); %wartość końca przedziału
while(sign (fa) == sign (fb)) %porównanie znaków wartości funkcji na końcach
printf ('Zły przedział, końce przedziałów muszą mieć różne znaki \n');
a=input('Nowy początek przedziału: ');      %początek przedziału
b=input('Nowy koniec przedziału: ');          %koniec przedziału
fa=f(a); %wartość początka przedziału
fb=f(b); %wartość końca przedziału
endwhile
i=0; % i to ilość wykonanych obrotów pętli
while(abs(b-a)>Eps) % sprawdzenie długości przedziału czy większy o epsilon
s = a - fa * (b - a) / (fb - fa); %wyliczenie punktu przecięcia siecznej z a i b
a=b; %z osią y=0 i zmienienie końców przedziałów na nowe
b=s;
fa=fb; % zmiana wartości funkcji w punktach na nowe
fb=f(s);
i=i+1; % dodanie jednego obrotu do pętli
endwhile
printf('Miejsce zerowe funkcji to: %.10f \n',s)   %wyświetlenie wyników
printf('Wartość funkcji w wyznaczonym miejscu zerowym to: %.10f \n',f(s))
printf('Ilość wykonanych obrotów pętli: %d \n',i)
endfunction
```

7 Testy na funkcjach

Przeprowadziliśmy testy 3 metod na wybranych przez nas funkcjach i wyniki przedstawiliśmy w tabelach.



7.1 funkcja $f(x) = \log_{10}(x) - 1$

7.1.1 metoda bisekcji

dla przedziału $[2; 20]$			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1		9.9453125000	- 0.0023815660
0.01	10	10.0068359375	0.0002967796
0.001	14	9.9996948242	- 0.0000132538
0.0001	17	10.000038470	0.000001657
0.00001	20	9.9999952316	- 0.0000002071
0.000001	24	10.0000000596	0.0000000026
0.0000001	27	9.9999999925	- 0.0000000003
0.00000001	30	10.0000000009	0.0000000000

dla przedziału $[1; 50]$			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	8	10.0917968750	- 0.0039685006
0.01	12	10.0020751953	0.0000901152
0.001	15	9.9998321533	- 0.0000072895
0.0001	18	9.9999256134	- 0.0000032306
0.00001	22	10.0000015497	0.0000000673
0.000001	25	9.9999993593	- 0.0000000278
0.0000001	28	9.9999999981	- 0.0000000001
0.00000001	32	10.0000000038	0.0000000002

dla przedziału $[9; 11]$			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	4	9.9375000000	- 0.0027228583
0.01	7	9.9921875000	- 0.0003394252
0.001	10	9.9990234375	- 0.0000424136
0.0001	14	9.9999389648	0.0000026507
0.00001	17	9.9999923706	- 0.0000003313
0.000001	20	9.9999990463	- 0.0000000414
0.0000001	24	9.9999999404	- 0.0000000026
0.00000001	27	9.9999999925	- 0.0000000003

7.1.2 Metoda siecznych

dla przedziału $[2; 20]$			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	5	9.9991512260	- 0.0000368634
0.01	6	10.0000017270	0.0000000750
0.001	6	10.0000017270	0.0000000750
0.0001	7	10.0000000001	0.0000000000
0.00001	7	10.0000000001	0.0000000000
0.000001	8	10.0000000000	0.0000000000
0.0000001	8	10.0000000000	0.0000000000
0.00000001	8	10.0000000000	0.0000000000

dla przedziału $[1; 50]$			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	13	9.9992593634	- 0.0000321519
0.01	14	10.0000110716	0.0000004808
0.001	15	10.0000000005	0.0000000000
0.0001	15	10.0000000005	0.0000000000
0.00001	16	10.0000000000	0.0000000000
0.000001	16	10.0000000000	0.0000000000
0.0000001	16	10.0000000000	0.0000000000
0.00000001	16	10.0000000000	0.0000000000

dla przedziału [9; 11]			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	2	9.9975376716	- 0.0001069507
0.01	3	10.0000061612	0.0000002676
0.001	4	10.0000000008	0.0000000000
0.0001	4	10.0000000008	0.0000000000
0.00001	4	10.0000000008	0.0000000000
0.000001	5	10.0000000000	0.0000000000
0.0000001	5	10.0000000000	0.0000000000
0.00000001	5	10.0000000000	0.0000000000

7.1.3 metoda Newtona

punkt startowy: 2			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	5	9.9999999869	- 0.0000000006
0.01	5	9.9999999869	- 0.0000000006
0.001	5	9.9999999869	- 0.0000000006
0.0001	6	10.0000000000	0.0000000000
0.00001	6	10.0000000000	0.0000000000
0.000001	6	10.0000000000	0.0000000000
0.0000001	6	10.0000000000	0.0000000000
0.00000001	7	10.0000000000	0.0000000000

punkt startowy: 30			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	10	10.0000000028	0.0000000028
0.01	10	10.0000000028	0.0000000028
0.001	11	10.0000000000	0.0000000000
0.0001	11	10.0000000000	0.0000000000
0.00001	11	10.0000000000	0.0000000000
0.000001	11	10.0000000000	0.0000000000
0.0000001	12	10.0000000000	0.0000000000
0.00000001	12	10.0000000000	0.0000000000

punkt startowy 12			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	3	9.9999998424	- 0.0000000068
0.01	3	9.9999998424	- 0.0000000068
0.001	4	10.0000000000	0.0000000000
0.0001	4	10.0000000000	0.0000000000
0.00001	4	10.0000000000	0.0000000000
0.000001	4	10.0000000000	0.0000000000
0.0000001	5	10.0000000000	0.0000000000
0.00000001	5	10.0000000000	0.0000000000

7.1.4 funkcja wbudowana fzero dla przedziału [2,20]

```
r =  10
fr = 0
z =  1
info =

scalar structure containing the fields:

iterations =  7
funcCount =  9
algorithm = bisection, interpolation
bracketx =

10   10

brackety =

0   0
```

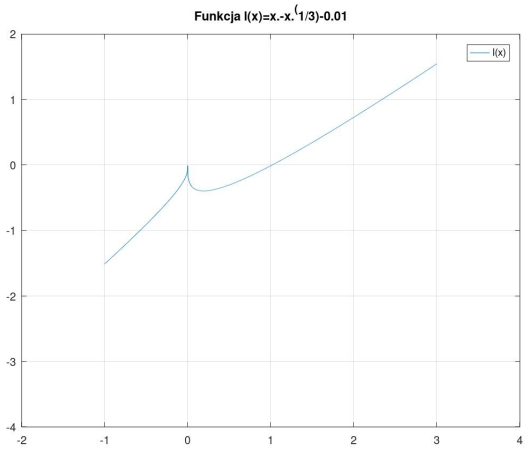
7.1.5 funkcja wbudowana fsolve dla punktu startowego 2

```
r = 10.0000
fr = -0.00000058178
z =  1
info =

scalar structure containing the fields:

iterations =  6
successful =  5
funcCount =  11
0   0
```

7.2 funkcja $f(x) = x - \sqrt[3]{x} - 0.01$



7.2.1 Metoda bisekcji

dla przedziału $[0; 2]$			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	4	1.0625000000	0.0320862245
0.01	7	1.0078125000	- 0.0047849143
0.001	10	1.0146484375	- 0.0002107253
0.0001	14	1.0149536133	- 0.0000062832
0.00001	17	1.0149612427	- 0.0000011719
0.000001	20	1.0149621964	- 0.0000005330
0.0000001	24	1.0149629712	- 0.0000000139
0.00000001	27	1.0149629936	0.0000000011

dla przedziału $[0; 20]$			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	7	1.0156250000	0.0004435604
0.01	10	1.0058593750	- 0.0060899477
0.001	14	1.0150146484	0.0000346076
0.0001	17	1.0149383545	- 0.0000165058
0.00001	20	1.0149669647	0.0000026616
0.000001	24	1.0149627924	- 0.0000001337
0.0000001	27	1.0149630159	0.0000000161
0.00000001	30	1.0149629880	- 0.0000000027

dla przedziału $[0; 100]$			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	9	1.0742187500	0.0400671617
0.01	13	1.0192871094	0.0028989682
0.001	16	1.0154724121	0.0003413146
0.0001	19	1.0149955750	0.0000218291
0.00001	23	1.0149657726	0.0000018629
0.000001	26	1.0149635357	0.0000003655
0.0000001	29	1.0149630718	0.0000000535
0.00000001	33	1.0149629961	0.0000000028

7.2.2 siecznych

dla przedziału $[0; 2]$			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	5	0.9854517484	- 0.0196751256
0.01	7	1.0149597387	- 0.0000021795
0.001	7	1.0149597387	- 0.0000021795
0.0001	8	1.0149629916	- 0.0000000002
0.00001	8	1.0149629916	- 0.0000000002
0.000001	9	1.0149629919	0.0000000000
0.0000001	9	1.0149629919	0.0000000000
0.00000001	9	1.0149629919	0.0000000000

dla przedziału $[0; 20]$			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	8	1.0147808653	- 0.0001220233
0.01	9	1.0149638092	0.0000005475
0.001	9	1.0149638092	0.0000005475
0.0001	10	1.0149629919	- 0.0000000000
0.00001	10	1.0149629919	- 0.0000000000
0.000001	11	1.0149629919	0.0000000000
0.0000001	11	1.0149629919	0.0000000000
0.00000001	11	1.0149629919	0.0000000000

dla przedziału $[0; 100]$			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	8	1.0145136354	0.0003011230
0.01	9	1.0149678785	0.0000032738
0.001	10	1.0149629912	- 0.0000000005
0.0001	10	1.0149629912	- 0.0000000005
0.00001	10	1.0149629912	- 0.0000000005
0.000001	11	1.0149629919	- 0.0000000000
0.0000001	11	1.0149629919	- 0.0000000000
0.00000001	11	1.0149629919	- 0.0000000000

7.2.3 metoda Newtona

punkt startowy: -2			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	4	1.0149642543	0.0000008457
0.01	4	1.0149642543	0.0000008457
0.001	5	1.0149629919	- 0.0000000006
0.0001	5	1.0149629919	- 0.0000000000
0.00001	5	1.0149629919	- 0.0000000000
0.000001	6	1.0149629919	0.0000000000
0.0000001	6	1.0149629919	0.0000000000
0.00000001	6	1.0149629919	0.0000000000

punkt startowy: 10			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	3	1.0150694214	0.0000713037
0.01	4	1.0149629938	0.0000000012
0.001	4	1.0149629938	0.0000000012
0.0001	5	1.0149629919	0.0000000000
0.00001	5	1.0149629919	0.0000000000
0.000001	5	1.0149629919	0.0000000000
0.0000001	5	1.0149629919	0.0000000000
0.00000001	5	1.0149629919	0.0000000000

punkt startowy 30			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	3	1.0157581932	0.0005328142
0.01	4	1.0149630943	0.0000000686
0.001	4	1.0149630943	0.00000000686
0.0001	5	1.0149629919	0.0000000000
0.00001	5	1.0149629919	0.0000000000
0.000001	5	1.0149629919	0.0000000000
0.0000001	6	1.0149629919	0.0000000000
0.00000001	6	1.0149629919	0.0000000000

7.2.4 funkcja wbudowana fzero dla przedziału [0,2]

```
r =  1.0150
fr =   -4.3542e-16
z =  1
info =

scalar structure containing the fields:

iterations =  8
funcCount =  10
algorithm = bisection, interpolation
bracketx =

1.0150   1.0150

brackety =

-4.3542e-16   8.6736e-18
```

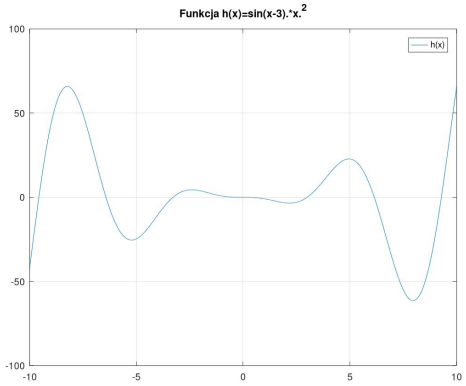
7.2.5 funkcja wbudowana fsolve dla punktu startowego 10

```
r =  1.0150
fr =  0.0000000012278
z =  1
info =

scalar structure containing the fields:

iterations =  5
successful =  4
funcCount =  9
```

7.3 funkcja $f(x) = sin(x-3)x^2$



7.3.1 bisekcji

dla przedziału $[8; 11]$			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	4	9.2187500000	- 5.4722686719
0.01	8	9.2832031250	0.0015355016
0.001	11	9.2824707031	- 0.0615733260
0.0001	14	9.2831115723	- 0.0063541918
0.00001	18	9.2831859589	0.0000561604
0.000001	21	9.2831852436	- 0.0000054786
0.0000001	24	9.2831853330	0.0000022263
0.00000001	28	9.2831853051	- 0.0000001815

7.3.2 siecznych

dla przedziału $[8; 11]$			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	3	9.2807585925	- 0.2090187468
0.01	4	9.2831678544	- 0.0015040337
0.001	5	9.28318516302	0.0000007881
0.0001	5	9.2831851630	0.0000007881
0.00001	6	9.2831853072	- 0.0000000000
0.000001	6	9.2831853072	- 0.0000000000
0.0000001	6	9.2831853072	- 0.0000000000
0.00000001	6	9.2831853072	- 0.0000000000

7.3.3 metoda Newtona

punkt startowy: 7			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	2	6.1416373039	- 0.0016841958
0.01	3	6.1415926543	- 0.0000000250
0.001	3	6.1415926543	- 0.0000000250
0.0001	3	6.1415926543	- 0.0000000250
0.00001	4	6.1415926536	0.0000000000
0.000001	4	6.1415926536	0.0000000000
0.0000001	4	6.1415926536	0.0000000000
0.00000001	4	6.1415926536	0.0000000000

punkt startowy: 8			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	4	46.9822971635	0.0000291237
0.01	4	46.9822971635	0.0000291237
0.001	4	46.9822971635	0.0000291237
0.0001	5	46.9822971503	- 0.0000000000
0.00001	5	46.9822971503	- 0.0000000000
0.000001	5	46.9822971503	- 0.0000000000
0.0000001	5	46.9822971503	- 0.0000000000
0.00000001	6	46.9822971503	- 0.0000000000

punkt startowy: 10			
ϵ (błąd)	l. pętli	x_0	$f(x_0)$
0.1	2	9.2833297066	0.0124443726
0.01	3	9.2831853117	0.0000003897
0.001	3	9.2831853117	0.0000003897
0.0001	4	9.2831853072	0.0000000000
0.00001	4	9.2831853072	0.0000000000
0.000001	4	9.2831853072	0.0000000000
0.0000001	4	9.2831853072	0.0000000000
0.00000001	4	9.2831853072	0.0000000000

7.3.4 funkcja wbudowana fzero dla przedziału [8,11]

```
r =  9.2832
fr =   -3.2727e-13
z =  1
info =

scalar structure containing the fields:

iterations =  8
funcCount =  10
algorithm = bisection, interpolation
bracketx =

9.2832   9.2832

brackety =

-3.2727e-13   2.8506e-13
```

7.3.5 funkcja wbudowana fsolve dla punktu startowego 10

```
r =  9.2832
fr =  0.00000038742
z =  1
info =

scalar structure containing the fields:

iterations =  4
successful =  3
funcCount =  7
```

8 zalety i wady poszczególnych metod

Metoda	zalety	wady
bisekcji	globalna zbieżność najsłabsze założenia zawsze zbieżna	najwolniejsza zbieżność
siecznych	najbardziej efektywna	zbieżność jedynie lokalna wymagające założenia wolna zbieżność
Newtona	najszybsza zbieżność	zbieżność jedynie lokalna wymagające założenia działania na pochodnych

9 Wnioski i podsumowanie

Na podstawie zaprogramowanych przez nas funkcji oraz funkcji wbudowanych służących do wyznaczania miejsc zerowych zauważamy, że wyznaczanie pierwiastków bez znajomości wyglądu funkcji jest niełatwe i często prowadzi do błędów.
W dwóch z trzech opisanych przez nas metod należy wyznaczyć badany przedział tak, aby wartości funkcji na jego końcach były różnych znaków, co powoduje kolejne trudności. Podsumowując, by przejść do szukania miejsca zerowego funkcji należy najpierw oszacować gdzie może się ono znajdować, bo szukanie na "chybił trafił" często prowadzi do błędnych wyników. Warto też dodać, że jednorazowo wyznaczymy tylko jedno miejsce zerowe. Jeśli podejrzewamy, że funkcja może mieć więcej pierwiastków, należy je szukać etapami przy pomocy zaprezentowanych w projekcie metod.
Dodatkowo zauważyliśmy również, że obieranie coraz to mniejszego błędu do przybliżenia szukanego przez nas rozwiązania skutkuje wykonaniem przez dany algorytm większej ilości obrotów pętli. Przy tym plusem jest, że pozyskany wynik będzie dokładniejszy. W projekcie przyjęta została dokładność do 10-go miejsca po przecinku, co widać na wykonanych przez nas zestawienich wyników w tabelkach. Wyniki te ukazały, że zmniejszanie błędu dla metody bisekcji poniżej 0.0001 niewiele zmienia w uzyskanym miejscu zerowym (zmiany rzędu 5-go miejsca po przecinku) za to ilość wykonanych obliczeń rośnie prawie dwukrotnie dla najmniejszej wartości błędu. Dlatego należy zastanowić się czy ważniejsza w danym przypadku jest dokładność czy szybkość zbieżności. Z kolei metoda Newtona i siecznych dla tych samych błędów czyli mniejszych od 0.0001 nie zmienia prawie nic i wyniki są prawie identyczne dla coraz to mniejszych błędów. Zarówno ilość wykonanych obliczeń jak i dokładność naszego miejsca zerowego rosnąc nie zmieniają się znacznie.
Zauważyliśmy, że ilość wykonanych obliczeń przez napisane przez nas algorytmy

i funkcje wbudowane są bardzo podobne.
Podsumowując, problem znajdowania miejsca zerowego funkcji metodami numerycznymi wymaga umiejętności ogólnego oszacowania ilości pierwiastków, a także ich położenia. Sporym minusem jest fakt, że obsługa skryptu nie będzie prosta dla przeciętnego użytkownika, ponieważ wymaga co najmniej ogólnej wiedzy na temat zarówno matematyki jak i metod numerycznych.

Literatura

[1] M. Kosiorowska, T. Stanisz: *Metody Numeryczne*

[2] A. Ralston: *Wstęp do Analizy Numerycznej*

[3] A. Ralston: *Wstęp do Metod Numerycznych*

[4] R. Zuber: *Metody Numeryczne i Programowanie*

[5] M. Dryja, J. Jankowska: *Przegląd metod i algorytmów numerycznych*

[6] Z. Kosma: *Metody i Algorytmy Numeryczne*

[7] R. Klemka, B.Świątek, A. Garbacz-Klempka: *Programowanie, Algorytmy Numeryczne i Modelowanie w Matlabie*

[8] W. Regel: *Obliczenia Symboliczne i Numeryczne w Programie Matlab*

[9] A. Bjorck, G.Dahlquist: *Metody Numeryczne*

[10] I. Gorgol materiały własne *Metody Numeryczne - konspekt wykładu dr Izolda Gorgol*

[11] serwis edukacyjny eduinf.waw.pl

[12] Rozwiązanie równania nieliniowego pl.wikibooks.org/wiki