Rozwiązywanie równań nieliniowych - omówienie zasadniczych problemów z przykladami

Opracowal: dr inz. Robert Jakubowski PRz

Do uruchomienia skryptu należy otworzyć go w Matlabie i włączyć zieliną strzałkę *Run*

W ustawieniach Matlaba prponuję zmienić wyświetlanie liczb z większą precyzją. Można to zrobi następująco:

Preferences - MATLAB - Command Window - Text display - Numeric format - Long

Table of Contents

Metody rozwiązyawania równań liniowych

Do rozwiazywania równań nieliniowych można wykorzystać metody:

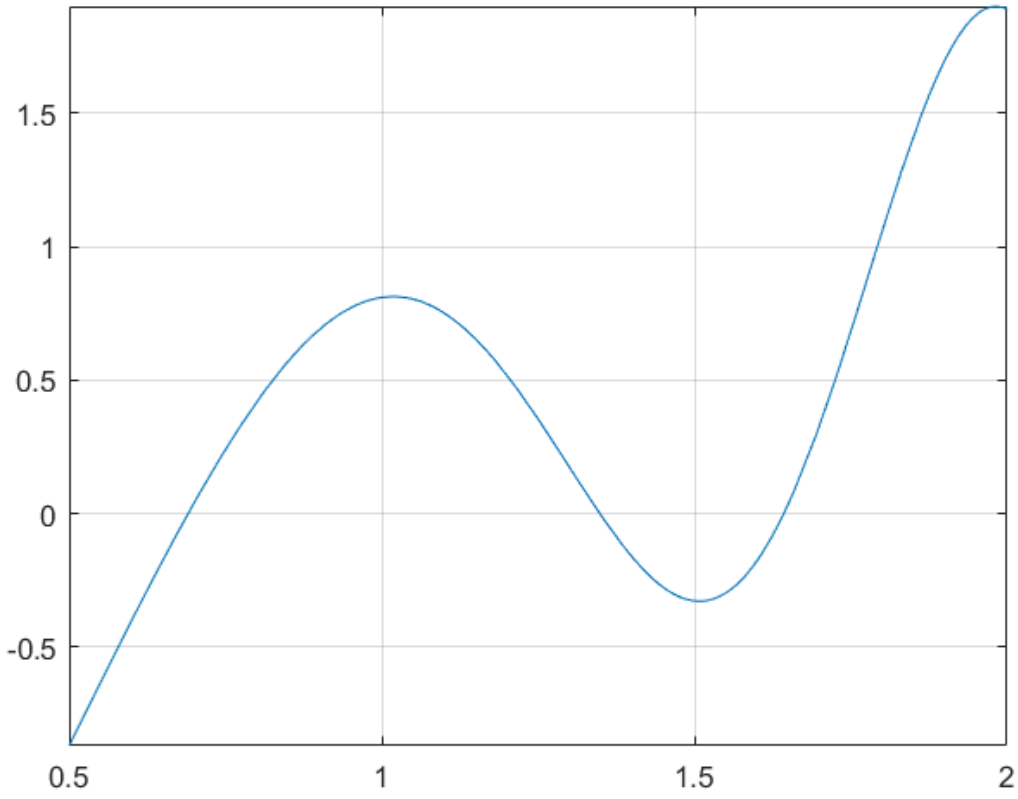
1. bisekcji/połowienia przedziału
2. metoda regula falsi
3. metoda siecznych
4. metoda stycznych / metoda Newtona

Metody 1, 2 wymagają zdefiniowania przedziału rozwiazywania w taki sposób, że rozwiązanie występuje pomiędzy jednym a drugim punktem podanego przedziału, czyli f(x1)*f(x2)<0.

Metody 3 i 4 nie wymagają zdefiniowania przedziału poszukiwania rozwiazywania tak, żeby rozwiązanie leżało pomiędzy podanymi punktami. W przypadku metody stycznych podaje się tylko jeden punkt początkowy, wykorzystywany do rozpoczęcia poszukiwania rozwiązania.

Możliwe podejścia do poszukiwania rozwiązania omawianymi metodami przedstawiają poniższe przykłady

```
% przygotowujemy funkcję zmiennej x
fx=@(x) sin(2.*x.^2)-(x-2).^2+0.9;
fplot(fx,[0.5 2.])

grid on
```



Jedno z rozwiązań funkcji f(x)=0 występuje pomiędzy 1 a 1.5. Przybliżenie tej funkcji w tym przedziale pokazuje wykres poniżej

```
fplot(fx,[1 1.5])
grid on
hold on
```

W **metodzie bisekcji** i **metodzie regula falsi** punktami początkowymi mogą być wartości x1=1 i x2=1.5.

Metoda bisekcji

W metodzie bisekcji x3 to połowa przedzialu x2 i x1 czyli:

$$x3 = \frac{(x1 + x2)}{2}$$

```
x1=1; x2=1.5;
x3=(x1+x2)/2;
X=[x1,x3,x2];
Y=fx(X);
plot(X,Y,'ro',[x3,x3],[0,Y(2)],'ro-');
text(x3-0.02,0-0.05,'x3_1')
disp(['wartosc f(x3_1) = ',num2str(Y(2))])
```

```
wartosc f(x3_1) = 0.35409
```

Sprawdzamy wartość otrzymanego rozwiązania. Jeżeli nie spełnia ono naszych oczekiwań dotyczących otrzymanego rozwiązania, prowadzimy dalsze obliczenia wg metody bisekcji. Sprawdźmy warunek dotyczący wyboru przedziału do następnego kroku. W tym celu wyliczymy wartość iloczynu f(x1)*f(x2). Jeżeli będzie to wartość większa od zera to znaczy, że do dalszego postępowania przyjmiemy przedział x3 - x2. Gdy wartość będzie mniejsza od zera to przedział x1 - x3.

```
iloczynf1f3=Y(1).*Y(2);
disp(['wartosc iloczynu = ',num2str(iloczynf1f3)])
```

```
wartosc iloczynu = 0.28657
```

Ponieważ wartość ta jest większa od zera zatem przypiszemy x1=x3 i powtórzymy obliczenia wg schematu z wcześniejszego kroku:

```
x1=x3;
x3=(x1+x2)/2;
X=[x1,x3,x2];
Y=fx(X);
plot(X,Y,'ro',[x3,x3],[0,Y(2)],'ro-');
text(x3-0.01,0+0.03,'x3_2')
disp(['wartosc f(x3_2) = ',num2str(Y(2))])
```

```
wartosc f(x3_2) = -0.087546
```

Jesteśmy bliżej poszukiwanego rozwiązania, ale jeżeli, ciągle nie jest ono dla nas zadawalające to powtórzymy operacje realizowane wcześniej, Sprawdzimy wartość iloczynu f(x1)*f(x3) i wybierzemy przedział do dalszego poszukiwania rozwiązania.

```
iloczynf1f3=Y(1).*Y(2);
disp(['wartosc iloczynu = ',num2str(iloczynf1f3)])
```

```
wartosc iloczynu = -0.030999
```

Ponieważ wartość iloczynu jest mniejsza od 0, zatem wybieramy tym razem przedział poszukiwania rozwiązania x1 - x3

```
x2=x3;
x3=(x1+x2)/2;
X=[x1,x3,x2];
Y=fx(X);
plot(X,Y,'ro',[x3,x3],[0,Y(2)],'ro-');
text(x3-0.02,0-0.05,'x3_3')
disp(['wartosc f(x3_3) = ',num2str(Y(2))])
```

```
wartosc f(x3_3) = 0.12827
```

Tym razem wartość f(x3) jest dalej od rozwiązania dokładnego niż uzyskana wcześniej. Iloczyn f(x1)*f(x3)>0, zatem rozwiązanie będzie dalej poszukiwane w przedziale x3 - x2

```
x1=x3;
x3=(x1+x2)/2;
X=[x1,x3,x2];
Y=fx(X);
plot(X,Y,'ro',[x3,x3],[0,Y(2)],'ro-');
```

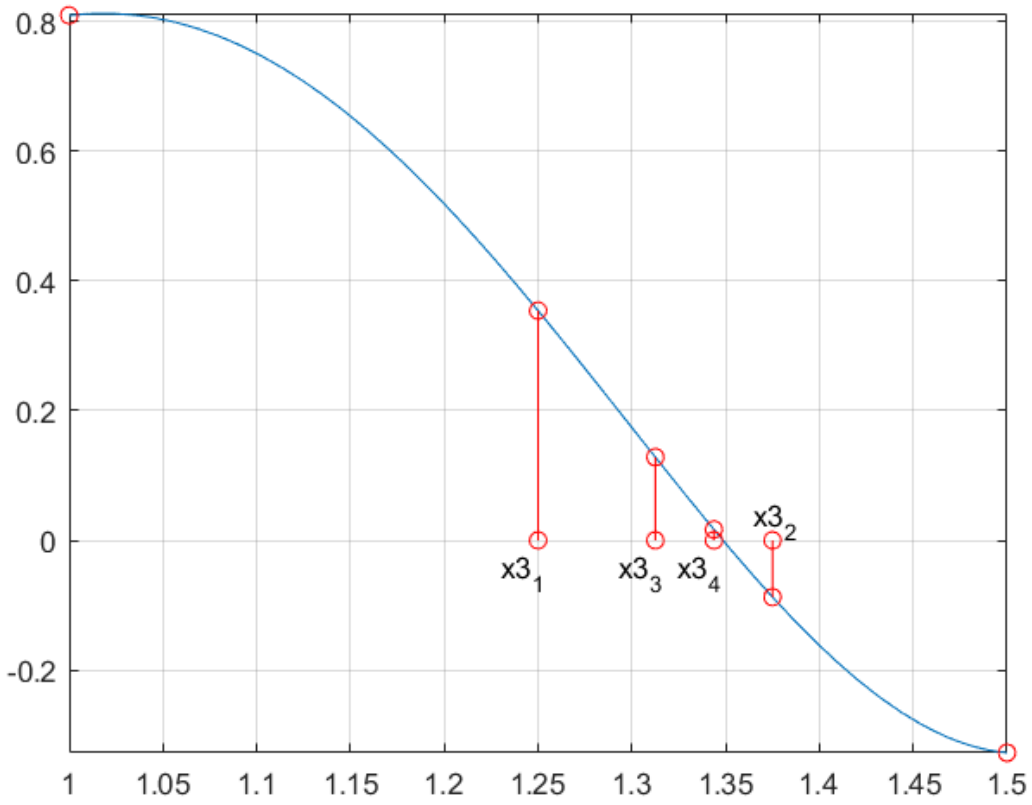
```
text(x3-0.02,0-0.05,'x3_4')
disp(['wartosc f(x3_4) = ',num2str(Y(2))])
```

```
wartosc f(x3_4) = 0.016686
```

Załóżmy że to rozwiązanie nas satysfakcjonuje wiec możemy zakończyć proces poszukiwania rozwiązania. Gdyby rozwiązanie to nie było satysfakcjonujące, to należałoby postępować dalej wg. pokazanego schematu.

Oczywiście proces ten można zautomatyzować. W funkcji metoda_bisekcji zawarto funkcje pełną metodę rozwiązania do momentu osiągnięcia zakładanego poziomu dokładności +-e.

```
hold off
```



Metoda regula falsi

Zasada postępowania wg tej metody będzie przedstawiona dla tego samego przedziału, jak metoda bisekcji, czyli w przedziale od 1 do 1.5

```
x1=1; x2=1.5;
fplot(fx,[0.95 1.55])
grid on
hold on
```

Pierwszy krok metody regula falsi przedstawiono pniżej, a x3 wyznaczono z zalezności:

$$x3 = \frac{x1 * y2 - x2 * y1}{y2 - y1}$$

```
x3=(x1*fx(x2)-x2*fx(x1))/(fx(x2)-fx(x1));
f3=fx(x3);
plot([x1,x3,x2],[fx(x1),0,fx(x2)],'go-',[x3,x3],[0,f3],'go-')
text(x3,0+0.015,'x3_1')
disp(['wartosc f(x3) = ',num2str(f3)])
```

```
wartosc f(x3) = -0.02515
```

Jeżeli uzyskane rozwiązanie nie spełnia założonego poziomu dokładności, to podobnie jak w metodzie bisekcji dokonujemy wyboru przedziału dalszego poszukiwania rozwiązania. Robimy to sprawdzając wartość iloczynu f(x1)*f(x3). Gdy jest on większy od zera to znaczy ze rozwiązanie będzie leżało w przedziale x3 - x2, gdy zaś iloczyn jest mniejszy od zera to rozwiązanie leży w przedziale x1 - x3.

W pokazanym przykładzie iloczyn f(x1)*f(x2)<0, zatem poszukiwane rozwiązania będzie w przedziale x1 - x3

```
iloczynf1f3=fx(x1).*f3;
disp(['wartosc iloczynu = ',num2str(iloczynf1f3)])
```

```
wartosc iloczynu = -0.020353
```

```
x2=x3;
x3=(x1*fx(x2)-x2*fx(x1))/(fx(x2)-fx(x1));
f3=fx(x3);
plot([x1,x3,x2],[fx(x1),0,fx(x2)],'ro-',[x3,x3],[0,f3],'ro-')
text(x3-0.015,0-0.05,'x3_2')
disp(['wartosc f(x3_2) = ',num2str(f3)])
```

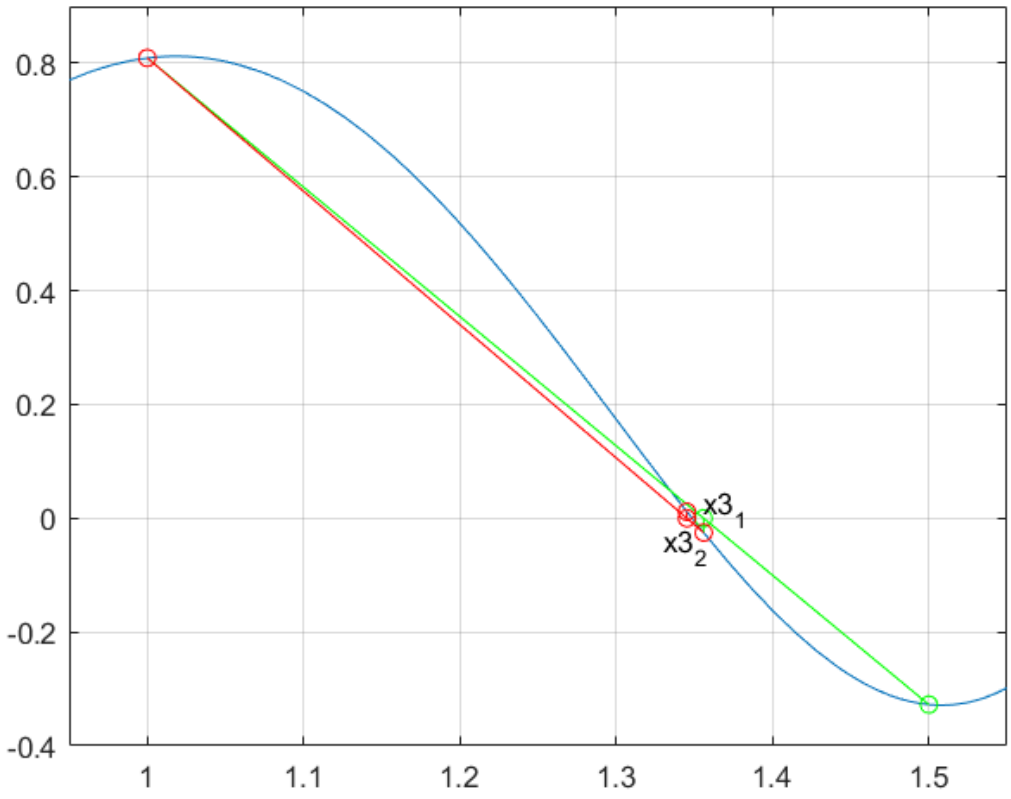
```
wartosc f(x3_2) = 0.011586
```

Przybliżmy rozwiązanie, aby zobaczyć wynik obliczeń. Użyj suwaka do skalowania wykresu

```
scala =1
```

```
scala =
     1
```

```
axis([0.95+(1-scala)*0.35,1.55-(1-scala)*0.15,-.4+(1-scala)*0.4,0.9-(1-scala)*0.9])
hold off
```



W przedstawionej metodzie już po dwóch krokach osiągnięto taki sam poziom dokładności rozwiązania, jak w metodzie bisekcji po czterech krokach. Metoda ta jest więc szybsza (szybciej zbieżna) niż metoda bisekcji

Metoda siecznych

W metodzie **siecznych**punkty początkowe rozwiązania mogą być położone jak w metodzie regula falsi, ale równie dobrze początkowe przybliżenie rozwiązania może być zdefiniowane następująco x1=1.1, x2=1.2. W tej metodzie nie ma warunku, aby przyjęty przedział poszukiwania był tak dobrany, aby rozwiązanie leżało pomiędzy przyjętymi punktami początkowymi poszukiwania. Poszukiwanie kolejnego punktu x3 prowadzi się wg zależności:

$$x3 = x2 - \frac{y2}{y2 - y1}(x2 - x1)$$

W następnym kroku x2 podstawia się za x1 zaś x3 przypisuje się za x2. Należy pamietać, aby w ten sam sposób postepować z y, czyli przypisać y1=y2 i y2=y3. W tej metodzie nie sprawdza sie w ktorym przedziale może lezeć rozwiazanie

Przyklad pierszego kroku metody siecznych przedstawiono kolorem zielonym

```
fplot(fx,[1 1.5])
grid on
hold on

x1=1.1; x2=1.2;
Y=fx([x1,x2]);
x3=x2-Y(2)/(Y(2)-Y(1))*(x2-x1);
```

```
f3=fx(x3);
plot([x1,x2,x3],[Y,0],'g<-',[x3,x3],[0,f3],'g<-')
text(x3,0-0.05,'x3_1')
disp(['wartosc f(x3_1) = ',num2str(f3)])
```

```
wartosc f(x3_1) = -0.22272
```

Kolejne dwa kroki zgodnie z opisem wygladą nastepujaco:

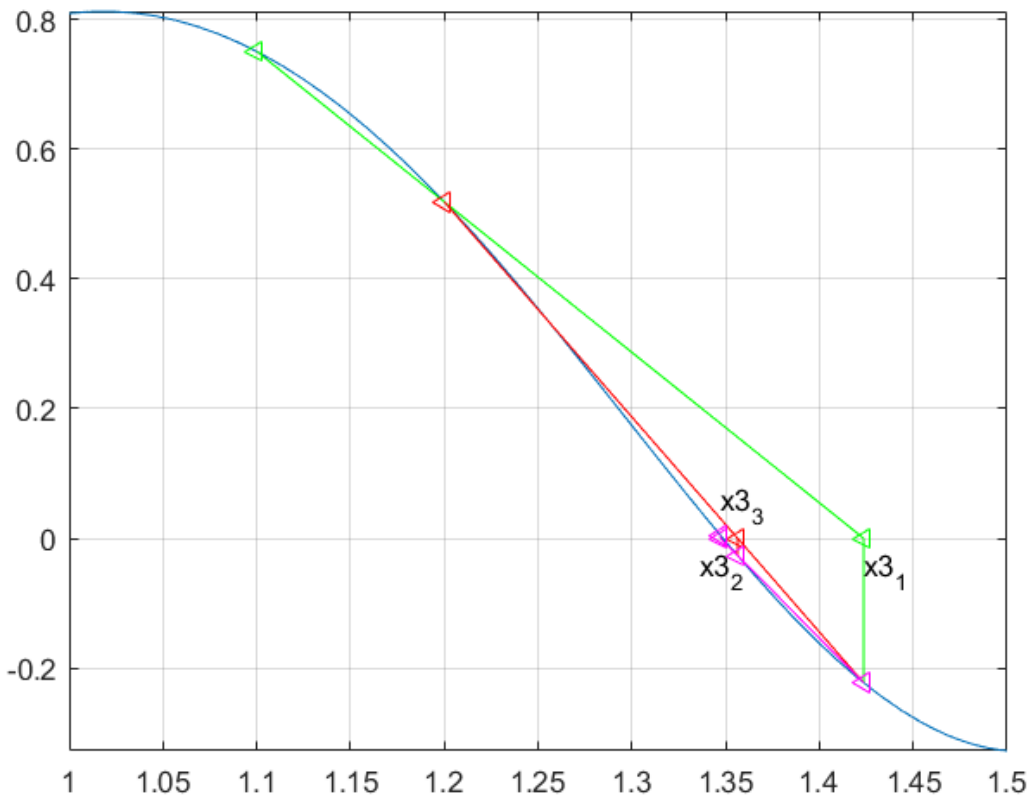
```
x1=x2; x2=x3;
Y=fx([x1,x2]);
x3=x2-Y(2)/(Y(2)-Y(1))*(x2-x1);
f3=fx(x3);
plot([x1,x2,x3],[Y,0],'r<-',[x3,x3],[0,f3],'r<-')
text(x3-0.02,0-0.05,'x3_2')
disp(['wartosc f(x3_2) = ',num2str(f3)])
```

```
wartosc f(x3_2) = -0.026713
```

```
x1=x2; x2=x3;
Y=fx([x1,x2]);
x3=x2-Y(2)/(Y(2)-Y(1))*(x2-x1);
f3=fx(x3);
plot([x1,x2,x3],[Y,0],'m<-',[x3,x3],[0,f3],'m<-')
text(x3,0+0.05,'x3_3')
disp(['wartosc f(x3_3) = ',num2str(f3)])
```

```
wartosc f(x3_3) = 0.0045353
```

```
hold off
```



Metoda dała zadawalające rozwiązanie po trzech krokach, podobnie jak metoda regula falsi

Metoda stycznych

W metodzie siecznych wykorzystuje się zalezność na poszukiwanie kolejnego punktu x2:

$$x2 = x1 - \frac{y1}{y1'}$$

W kolejnych krokach poszukiwania rozwiazania postepuje sie podobnie jak w metodzie siecznych, czyli w kolejnych krokach x2 zastepuje x1. Podobnie zastępuje się wartości y.

Pochodną funkcji można przyblizyć ilorazem różnicowym, czyli:

$$y1' = \frac{y(x1 + dx) - y(x1)}{dx}$$

gdzie dx można przyjąć jako odpowiednio mały przyrost wartości x np. dx=1e-6;

Przyjmując x1=1.1 pierwszy krok poszukiwania rozwiązania prezentują graficznie punkty i linie w kolorze fioletowym

```
fplot(fx,[1 1.7])
grid on
hold on

x1=1.1;  dx=1e-6;
y1=fx(x1);
```

```
y1p=(fx(x1+dx)-y1)/dx;
x2=x1-fx(x1)/y1p;
y2=fx(x2);
plot([x1,x2,x2],[y1,0,y2],'color','m','LineStyle','-',...
    'Marker','diamond','LineJoin','chamfer')
text(x2-0.015,0+0.02,'x2_1')
disp(['wartosc f(x2_1) = ',num2str(y2)])
```

```
wartosc f(x2_1) = -0.18047
```

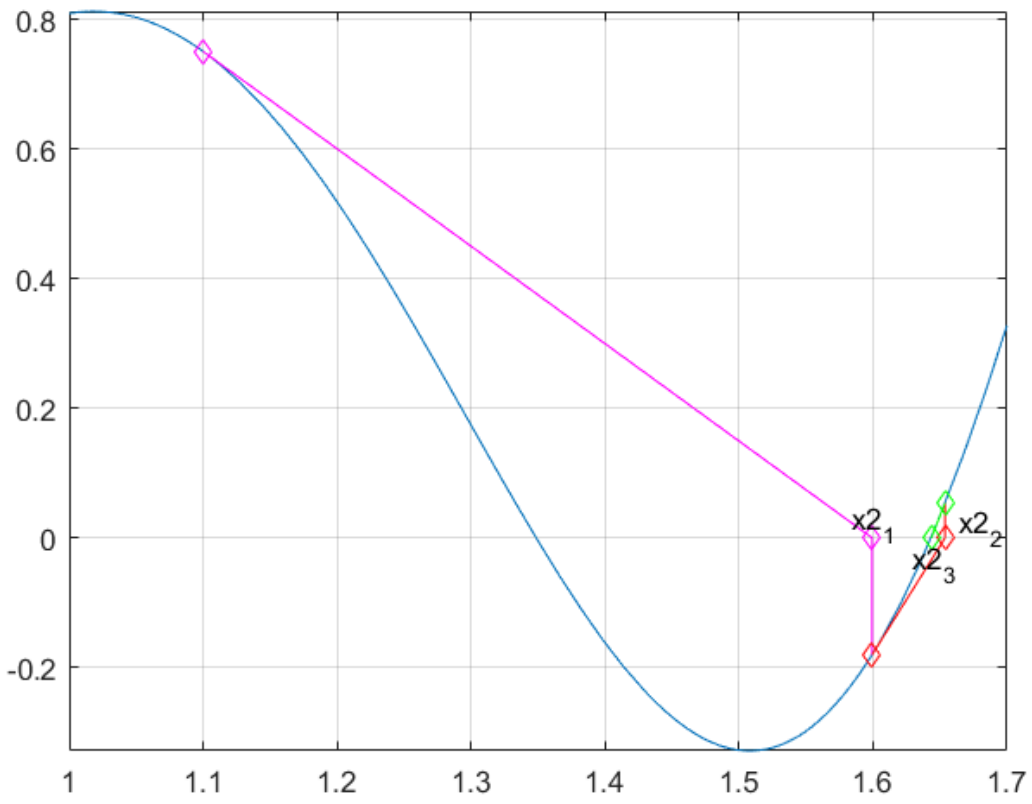
```
%krok 2
x1=x2;
y1=fx(x1);
y1p=(fx(x1+dx)-y1)/dx;
x2=x1-fx(x1)/y1p;
y2=fx(x2);
plot([x1,x2,x2],[y1,0,y2],'color','r','LineStyle','-',...
    'Marker','diamond','LineJoin','chamfer')
text(x2+0.01,0+0.015,'x2_2')
disp(['wartosc f(x2_2) = ',num2str(y2)])
```

```
wartosc f(x2_2) = 0.05372
```

```
%krok 3
x1=x2;
y1=fx(x1);
y1p=(fx(x1+dx)-y1)/dx;
x2=x1-fx(x1)/y1p;
y2=fx(x2);
plot([x1,x2,x2],[y1,0,y2],'color','g','LineStyle','-',...
    'Marker','diamond','LineJoin','chamfer')
text(x2-0.015,0-0.04,'x2_3')
disp(['wartosc f(x2_3) = ',num2str(y2)])
```

```
wartosc f(x2_3) = 0.0017353
```

```
hold off
```



Tym razem otrzymane rozwiązanie jest w innym punkcie niż w poprzednich przykładach. Podobna sytuacja mogła wystąpić dla metody siecznych. Obydwie metody poszukują rozwiązania bez ograniczonego przedziału przeszukiwania, jak w dwóch pierwszych metodach. Dlatego w zależności od podania punktu początkowego w prezentowanych metodach można uzyskać różne rozwiązania, niekoniecznie najbliższe zdefiniowanym punktom początkowym.

Podsumowanie

Poszukiwanie rozwiązania metodą bisekcji i regula falsi wymaga określenia początkowego przybliżenia w taki sposób, aby rozwiązanie leżało pomiędzy przyjętymi punktami x1 i x2.

Metoda siecznych wymaga podania dwóch punktów jako początkowe rozwiązanie, ale nie ma wymogu aby rozwiązanie znajdowało się pomiędzy tymi punktami.

W metodzie Newtona/stycznych do rozwiązania wskazuje się jeden punkt początkowy.

W metodzie bisekcji i regula falsi x3 podstawia sią za x1 lub x2 w zależności od tego w którym przedziale występuje rozwiązanie.

W metodzie siecznych i stycznych x2 podstawia się za x1, a x3 za x2 (metoda siecznych), nie sprawdza się przedziałów występowania rozwiązania

Przykłady rozwiązania zadań wymienionymi metodami

Przyjrzyjmy się jak będzie wyglądało rozwiązanie dla wskazanego przybliżenia początkowego dla metody bisekcji, stycznych i siecznych. Metodę regula falsi możecie dodać samodzielnie wstawiając na dole funkcję regula falsi opracowaną w ramach zadania nr 1. Następnie w kodzie programu należy dodać dodatkowe linie kodu.

Załóżmy, że wszystkie rozwiązania będą poszukiwane z dokładnością e=1e-6. Przyjrzyjmy się ilościom kroków iteracji oraz uzyskanym wartościom f(x) poszukiwanego rozwiązania

Rozwiązanie dla metody bisekcji - przedstawiono punktem w kolorze zielonym

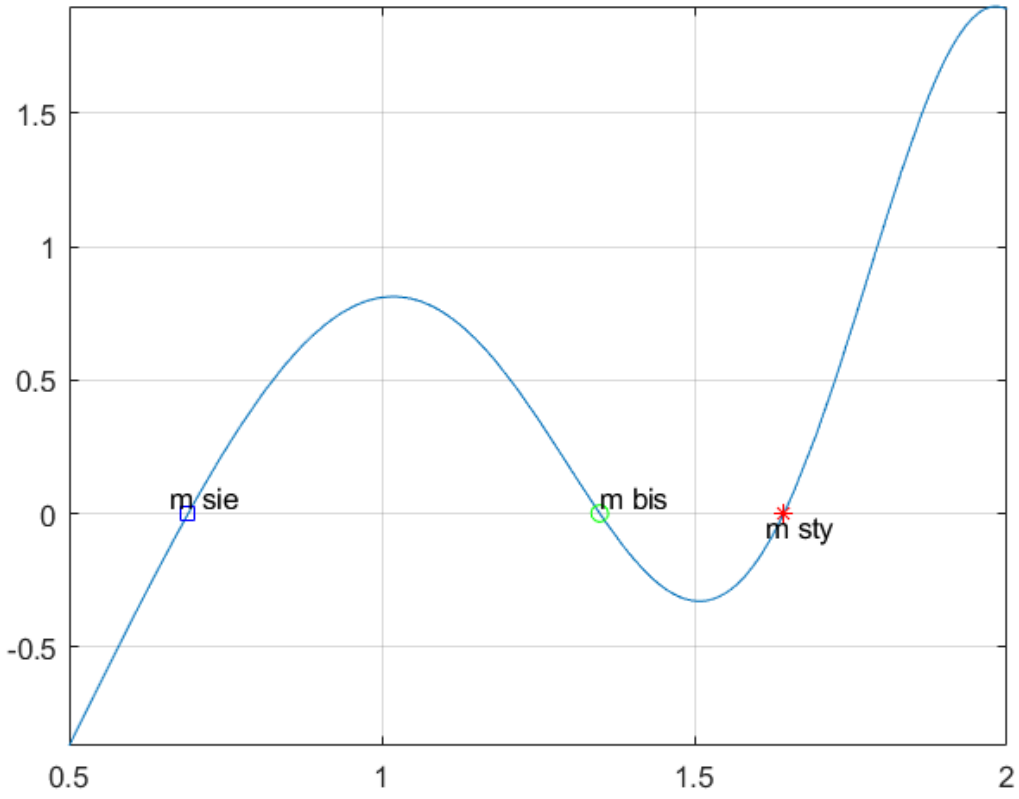
```
fplot(fx,[0.5 2])
grid on
hold on

x1=1; x2=1.5;  e=1e-6;
[x0_mb,y0_mb,n_mb]=metoda_bisekcji(fx,x1,x2,e);
plot(x0_mb,y0_mb,'og')
text(x0_mb,y0_mb+0.06,'m bis')
```

Rozwiązanie dla metody siecznych przedstawiono kwadratem w kolorze niebieskim

```
x1=1; x2=1.2;
[x0_ms,y0_ms,n_ms]=metoda_siecznych(fx,x1,x2,e);
plot(x0_ms,y0_ms,'sb')
text(x0_ms-0.03,y0_ms+0.06,'m sie')
```

Rozwiązanie dla metody stycznych przedstawiono w czerwoną gwiazdką

```
x1=1.1;
[x0_mst,y0_mst,n_mst]=metoda_stycznych(fx,x1,e);
plot(x0_mst,y0_mst,'*r')
text(x0_mst-0.03,y0_mst-0.05,'m sty')
hold off
```



Uzyskane wyniki pokazują, że każda z metod dała inne rozwiązanie. Ponadto metoda siecznych poprzezniecо inny dobór przybliżenia początkowego niż poprzednio, dała inne rozwiązanie. Przyjrzyjmy sie uzyskanym rezultatom:

```
disp(['x0 met. bisekcji = ',num2str(x0_mb)])
```

```
x0 met. bisekcji = 1.3486
```

```
disp(['f0 met. bisekcji = ',num2str(y0_mb)])
```

```
f0 met. bisekcji = -4.5937e-07
```

```
disp(['x0 met. siecznych = ',num2str(x0_ms)])
```

```
x0 met. siecznych = 0.69033
```

```
disp(['f0 met. siecznych = ',num2str(y0_ms)])
```

```
f0 met. siecznych = 1.2098e-08
```

```
disp(['x0 met. stycznych = ',num2str(x0_mst)])
```

```
x0 met. stycznych = 1.6432
```

```
disp(['f0 met. stycznych = ',num2str(y0_mst)])
```

```
f0 met. stycznych = 3.1897e-12
```

Przyglądając się wynikom widzimy, że metoda bisekcji dala rozwiązanie na poziomie dokładności poniżej

+-1e-7, metoda siecznych +-1e-8 a metoda stycznych +-1e-12, chociaż w każdym przypadku zadana dokładność rozwiązania była na poziomie +-1e-6.

Popatrzmy na ilość kroków iteracji potrzebnych do osiągniecia rozwiązania dla poszczególnych metod:

```
disp(['ilosc krokow met. bisekcji  = ',num2str(n_mb)])
```

```
ilosc krokow met. bisekcji  = 20
```

```
disp(['ilosc krokow met. siecznych = ',num2str(n_ms)])
```

```
ilosc krokow met. siecznych = 12
```

```
disp(['ilosc krokow met. stycznych = ',num2str(n_mst)])
```

```
ilosc krokow met. stycznych = 5
```

W przypadku metody bisekcji potrzebne było 20 kroków do otrzymania rozwiązania, podczas gdy metoda siecznych osiągnęła rozwiązanie na określnym poziomie po 12, a metoda stycznych po 5 krokach. Pokazuje to, która z metod (metoda stycznych) pozwala najszybciej osiągnąć rozwiązanie

Podsumowanie

Rozwiązanie uzyskane metodą bisekcji występuje w oczekiwanym przedziale nakreślonym podczas definiowana obszaru poszukiwanego rozwiązania. W przypadku pozostałych dwóch metod rozwiązanie wyznaczono w innych punktach, niż zakładano je otrzymać.

Pokazuje to, że wykorzystanie tych metod wymaga definiowanie punktów startowych blisko oczekiwanego rozwiązania, w przeciwnym wypadku można uzyskać inne niż oczekiwane rozwiązanie.

Metoda siecznych pozwala otrzymać rozwiązanie z bardzo dużą dokładnością w stosunkowo małej ilości kroków, podczas gdy pozostałe metody wymagają znacznie większej ilości kroków. Najwolniej zbieżną metodą jest metoda bisekcji.

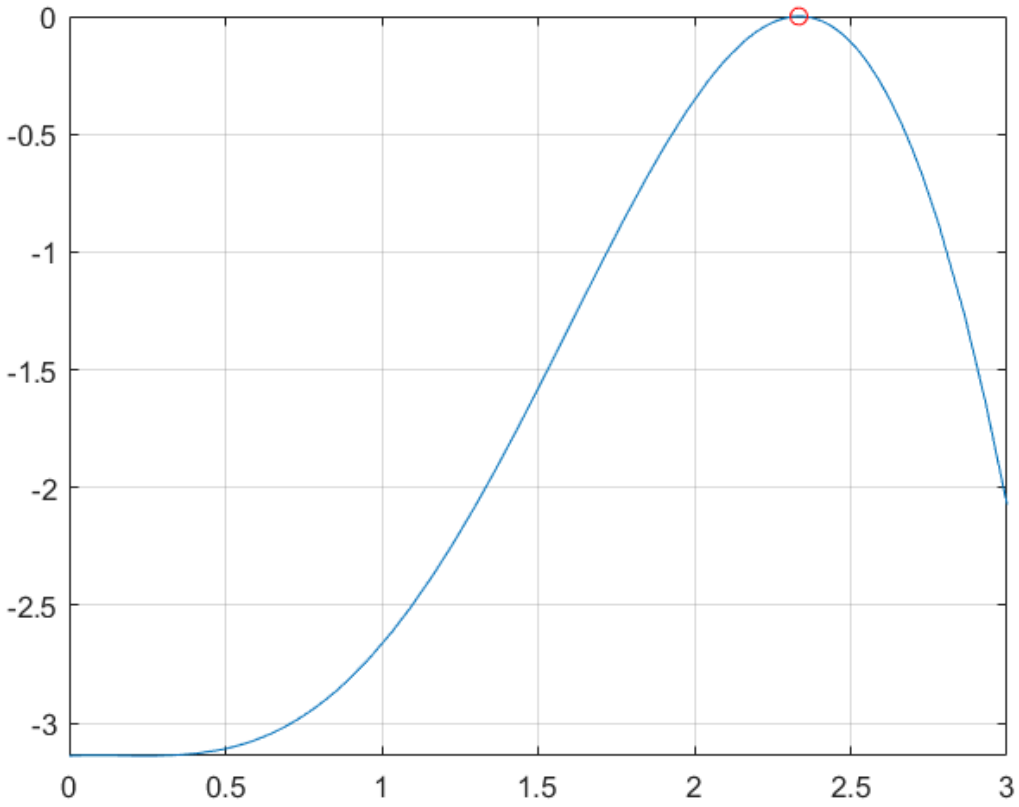
Rozwiazania dla funkcji, gdy f(x)=0 jest zarazem extremum funkcji.

Metody siecznych i stycznych ze względu na to, że nie wymagają określenia przedziału dla którego spełniony jest warunek f(x1)*f(x2)<0 pozwalają na poszukiwanie rozwiązania, gdy jest ono zarazem lokalnym ekstremum funkcji patrz przykład

```
fx=@(x) (x-0.25011).^2.*sin(x)-3.138072322414754;
fplot(fx,[0,3])
grid on
hold on
```

Przedstawiona funkcja pokazuje ze pomiedzy 2 a 2.5 występuje rozwiazanie i jest to ekstremum lokalne. Do rowiązania tego zadania możemy wykorzystać jedynie metodę stycznych lub siecznych. Przyjmijmy że wykorzystamy metodę stycznych, definujac punkt poczatkowy rozwiazania jako x1=2

```
x1=2;
[x0_mst,y0_mst,n_mst]=metoda_stycznych(fx,x1,e);
plot(x0_mst,y0_mst,'ro')
hold off
```



Wyniki rozwiązania

```
disp(['x0 = ',num2str(x0_mst)])
```

```
x0 = 2.3342
```

```
disp(['y0 = ',num2str(y0_mst)])
```

```
y0 = -3.0541e-07
```

```
disp(['ilosc krokow = ',num2str(n_mst)])
```

```
ilosc krokow = 9
```

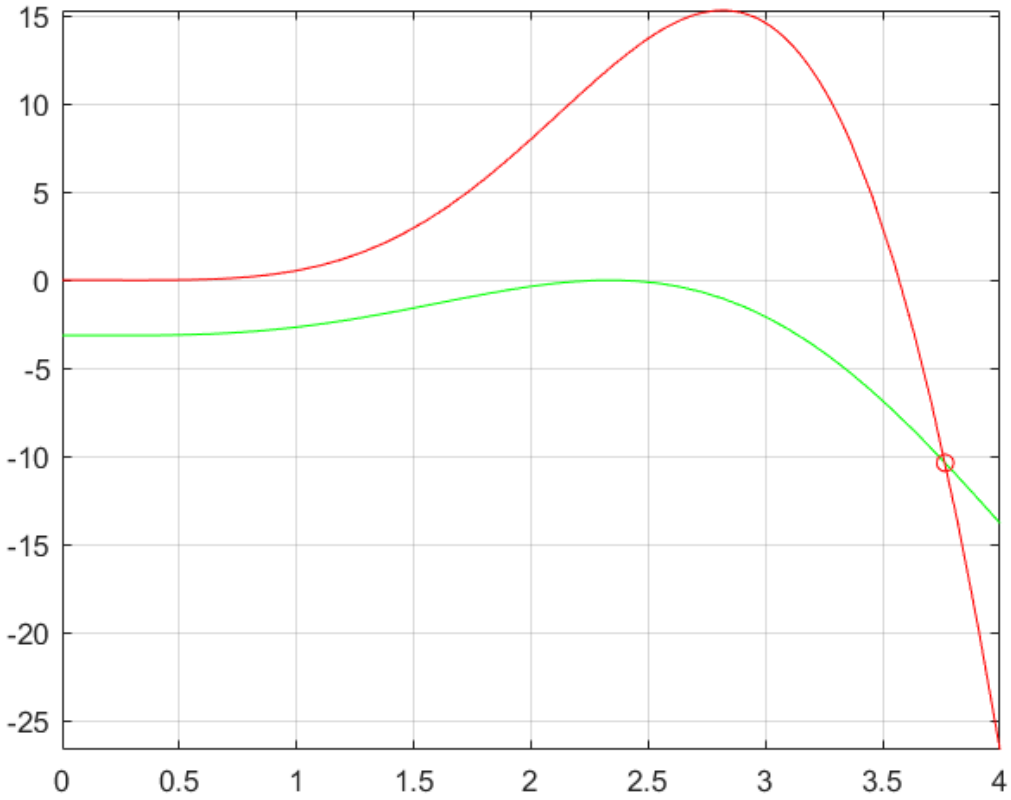
Poszukiwanie rozwiazania rownosci funkcji f(x)=g(x)

Pokazane metody mogą być wykorzystane do poszukiwania rozwiązania dla określenia równości funkcji f(x)=g(x). Zapiszmy dwie funkcje f(x) i g(x) i przedstawmy je graficznie na wykresie.

```
fx=@(x) (x-0.25011).^2.*sin(x)-3.138072322414754;
gx=@(x) cos(x-2).*x.^3;
x1=0; x2=4;
fplot(fx,[x1,x2],'g')
hold on
fplot(gx,[x1,x2],'r')
grid on
```

Przedstawione graficznie wykresy pokazują, że rozwiązanie występuje pomiędzy 3.5 a 4. Aby znaleźć dokładną wartość rozwiązania należy stworzyć nową funkcję poprzez odjęcie jednej funkcji od drugiej. Następnie jedną ze znanych metod znaleźć rozwiązanie poprzez przyrównanie nowej funkcji do 0.

```
kx=@(x) fx(x) - gx(x);
[x0]=metoda_bisekcji(kx,3.5,4,1e-6);
f0=fx(x0);
g0=gx(x0);
plot(x0,f0,'or')
hold off
```




```
disp(['x0 = ',num2str(x0)])
```

```
x0 = 3.7661
```

```
disp(['f0 = ',num2str(f0)])
```

```
f0 = -10.3662
```

```
disp(['g0 = ',num2str(g0)])
```

```
g0 = -10.3662
```

Funkcje f(x) i g(x) dla wyznaczinej wartosci x0 osiagają te same wartości, znaczy że x0 jest poszukiwanym roziwązaniem.

Wykorzystanie funkcji Matlaba do rozwiązywania rwónań nieliniowych jednej zmiennej

W Matlabie do rozwiązywania zadania mamy dostępną funkcje fzero. Aby się z nią zapoznać można wykorzystać funkcje help w Matlabie w postaci:

help fzero

wtedy dostaniemy informacje o funkcji fzero.

Typowe wywołanie funkcji to:

X = fzero(FUN,X0)

X jest poszukiwanym rozwiązaniem, FUN - funkcja, a X0 początkowym przybliżeniem rozwiązania, które może być jedno, lub dwuelementowe.

Sprawdźmy to na przykładzie.

```
fx=@(x) sin(2.*x.^2)-(x-2).^2+0.9;
fplot(fx,[0.5 2.])
hold on
grid on
x0=fzero(fx,1);
```

rezultatem w tym wypadku jest warosc xo

```
disp(['x0 = ',num2str(x0)])
```

```
x0 = 0.69033
```

```
plot(x0,0,'ro')
```

Funkcje te mozna takze wywolac w postaci:

[X,Y]=fzero(FUN,XO)

W tym wypadku od razu dostaje sie miejsce wystepowania rozwiazania x0 i y0

```
[x0,y0]=fzero(fx,1);
disp(['x0 = ',num2str(x0)])
```

```
x0 = 0.69033
```

```
disp(['y0 = ',num2str(y0)])
```

```
y0 = -4.4409e-16
```

Można też wywołać tą funkcję podając przedział poszukiwanego rozwiązania w postaci:

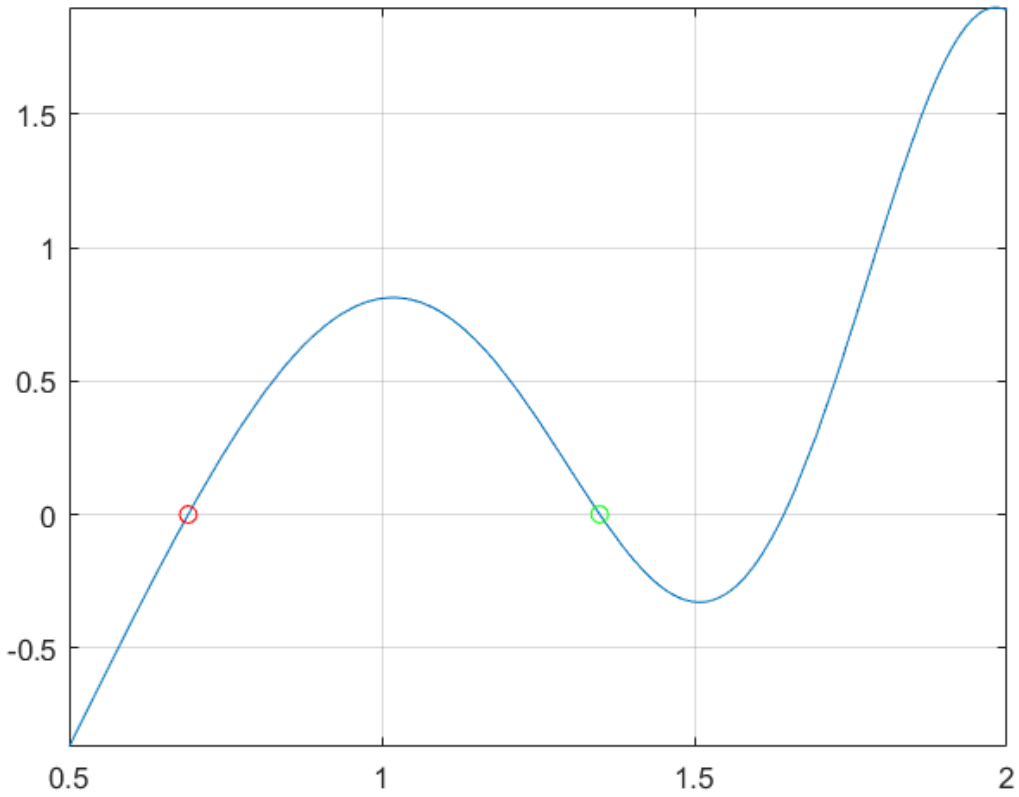
```
[x0,y0]=fzero(fx,[1, 1.5]);
disp(['x0 = ',num2str(x0)])
```

```
x0 = 1.3486
```

```
disp(['y0 = ',num2str(y0)])
```

```
y0 = -4.4409e-16
```

```
plot(x0,y0,'go')
hold off
```



Na koniec sprawdźmy, co otrzymamy gdy zdefiniujemy przedział, w którym nie ma rozwiązania.

```
%[x0,y0]=fzero(fx,[1, 1.2]);
%disp(['x0 = ',num2str(x0)])
%disp(['y0 = ',num2str(x0)])
```

Wynik daje błędne rozwiązanie o czym można się przekonać odblokowując działanie powyższych linii - usuwając znak %.

Miejsce na samodzielne cwiczenia

```

```

Funkcje przygotowane do obliczeń

function [x0,f0,n]=metoda_bisekcji(f,x1,x2,e)

```
function [x0,f0,n]=metoda_bisekcji(f,x1,x2,e)

%Funkcja rozwiązuje równanie f(x)=0 w przedziale od x1 do x2. e jest
%wskaźnikiem charakteryzującym dokładność rozwiązania, gdy nie jest zadany
%przyjmuje wartość e=1e-6
%Wartościami wyjściowymi są
% x0 - wartość rozwiązania (x dla którego f(x)=0)
% f0  - wartość rozwiązania f=0+-e
% n - ilość kroków

% Warunke sprawdzający
if exist('e','var')==0
    e=1e-6;
end
% Wyznaczenie wartości funkcji od x1
f1=f(x1);
% Wyznaczenie wartości funkcji od x2
f2=f(x2);
% Ustawienie wartości licznika do liczenia ilości kroków iteracji
n=0;


if abs(f1)<e
    % w tym przypadku x1 jest rozwiązaniem
    x0=x1;
    f0=f1;
elseif abs(f2)<e
    % w tym przypadku x2 jest rozwiązaniem
    x0=x2;
    f0=f2;
elseif f1*f2>0
    % w tym przypadku brakuje rozwiązania
    f0=NaN;
    x0=NaN;
else
    % w tym przypadku poszukiwane jest rozwiązanie
    while abs(f1)>e && abs(f2)>e
        n=n+1;
        %Poszukiwanie rozwiązania metodą połowienia przedziału
        x3=(x2+x1)/2;
        f3=f(x3);

        if f1*f3<0
            x2=x3;
            f2=f3;
        else
            x1=x3;
            f1=f3;
        end
```

```
    end
    %Przypisanie rozwiązania do zmiennych wyjściowych
    x0=x3;
    f0=f3;
end
end
```

function [x0,f0,n]=metoda_siecznych(fx,x1,x2,e)

```
function [x0,y0,n]=metoda_siecznych(fx,x1,x2,e)

%Function solve equation fx0=fx=0, where fx - fynction of x, fx=@(x) x..
%x1, x2 - start points of method, e- accuracy of solving

f1=fx(x1);
f2=fx(x2);
n=0;
if abs(f1)<=e
    x0=x1;
    y0=f1;
elseif abs(f2)<=e
    x0=x2;
    y0=f2;

else

    n=n+1;
    x=x2-f2/(f2-f1)*(x2-x1);
    %y=eval(fx);
    y=fx(x);
    while (abs(y)>e) && (n<100)
        x1=x2;
        f1=f2;
        x2=x;
        f2=y;
        x=x2-f2/(f2-f1)*(x2-x1);
        %y=eval(fx);
        y=fx(x);
        n=n+1;
    end
    if n==100
        x0=NaN;
        y0=NaN;
    else
        x0=x;
        y0=y;
    end
end
end
```

function [x0,fx0,n]=metoda_stycznych(fx,x1,e)

```
function [x0,fx0,n]=metoda_stycznych(fx,x1,e)

%Function solve equation fx0=fx=0, where fx - fynction of x, fx=@(x) x..
%x1 - start point of Nevton method, e- accuracy of solving
```

```
dx=0.00001;

f1=fx(x1);
xdx=x1+dx;
f2=fx(xdx);
x3=x1-(f1*dx)/(f2-f1);
f3=fx(x3);
n=1;
while ((abs(f3)>e) && (n<100))
    n=n+1;
    dx=abs(0.00001*(x3-x1));
    x1=x3;
    f1=fx(x1);
    xdx=x1+dx;
    f2=fx(xdx);
    x3=x1-(f1*dx)/(f2-f1);
    f3=fx(x3);
end
if n>=100
    x0=NaN;
    fx0=NaN;
else
    x0=x3;
    fx0=f3;
end
end
```