

4. Rešavanje nelinearnih jednačina

Data je funkcija:

$$f(x) = \sin x$$

1. Nacrtati funkciju na intervalu $x \in \left[\frac{\pi}{3}, \frac{4\pi}{3}\right]$ i zadržati grafik:

```
f = @(x) sin(x);  
  
x = linspace(pi/3, 4*pi/3, 100);  
fX = f(x);  
plot(x, fX, 'blue'), hold on
```

2. Nacrtati x -osu na intervalu $x \in \left[\frac{\pi}{3}, \frac{4\pi}{3}\right]$:

```
plot([pi/3 4*pi/3], [0 0], 'black')
```

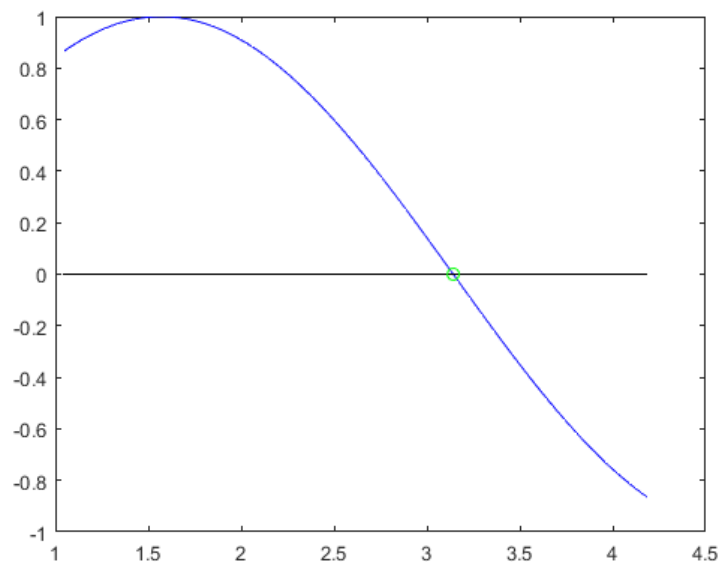
3. Pronaći i nacrtati **nulu funkcije** (isključiti zadržavanje grafika). Početi traženje sa **kraja intervala**:

```
zero = fzero(f, 4*pi/3)  
fZero = f(zero)  
scatter(zero, fZero, 'green'), hold off
```

Rezultat:

```
zero =  
    3.1416
```

```
fZero =  
    1.2246e-16
```



Slika 1. Nula funkcije

1. Metoda polovljenja

Zadatak 1. Napisati metodu polovljenja za traženje nule nelinearnih funkcija.

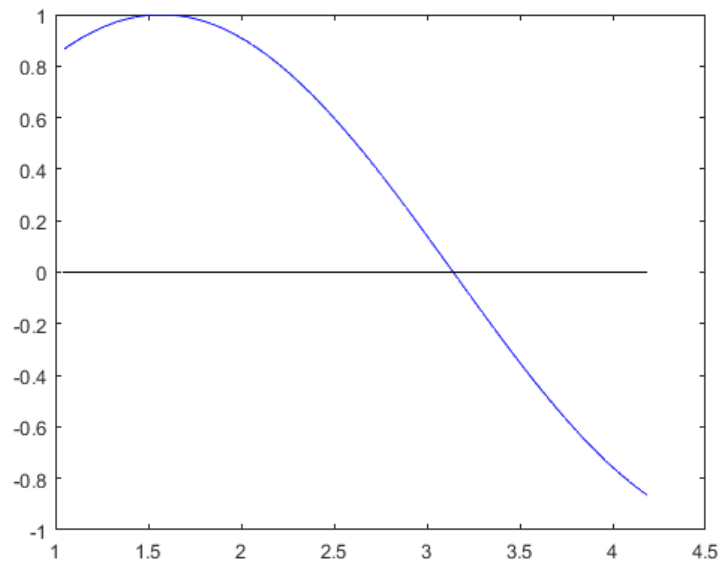
Pokušati prvo ručno jednu iteraciju metode polovljenja nad funkcijom $f(x) = \sin x$ na intervalu $x \in \left[\frac{\pi}{3}, \frac{4\pi}{3}\right]$:

```
f = @(x) sin(x);  
a = pi/3;  
b = 4*pi/3;
```

1. Nacrtati funkciju nad intervalom i naći njen minimum i maksimum i zadržati grafik:

```
x = linspace(a, b, 100);  
fX = f(x);  
fMin = min(fX);  
fMax = max(fX);  
plot(x, fX, 'blue', [a b], [0 0], 'black'), hold on
```

Rezultat:

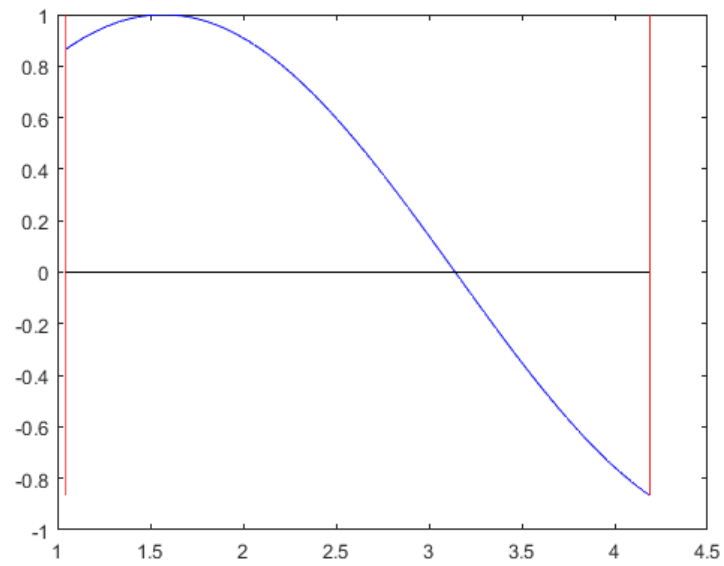


Slika 2. Grafik funkcije

2. Nacrtati 2 vertikalne ose na početku i na kraju intervala crvenom bojom:

```
plot([a a], [fMin fMax], 'red', [b b], [fMin fMax], 'red')
```

Rezultat:



Slika 3. Interval

3. Pretpostaviti nulu funkcije na polovini intervala i izračunati vrednost funkcije u pretpostavljenoj nuli:

```
zero = (a + b) / 2;  
fZero = f(zero)
```

Rezultat:

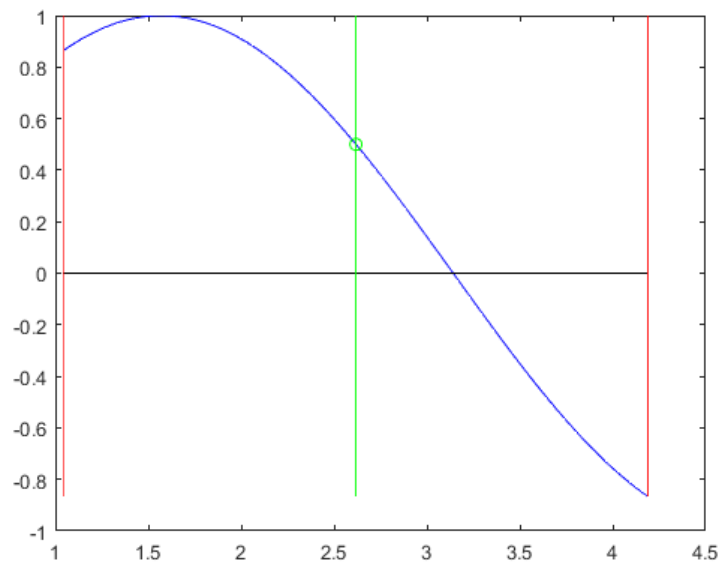
```
fZero =  
    0.5000
```

Izračunata vrednost je različita od 0.

4. Nacrtati vertikalnu osu i tačku određenu izračunatom vrednošću u pretpostavljenoj nuli funkcije zelenom bojom:

```
plot([zero zero], [fMin fMax], 'green', zero, fZero, 'o green')
```

Rezultat:



Slika 4. Pretpostavljena nula

5. Na osnovu izračunate vrednosti funkcije u jednom od krajeva intervala, pripremiti podinterval za narednu iteraciju. Odabrati onaj od dva podintervala($[a, zero]$ ili $[zero, b]$) na čijim krajevima vrenost funkcije ima različit znak:

```
if f(a)*fZero < 0
    b = zero;
else
    a = zero;
end
```

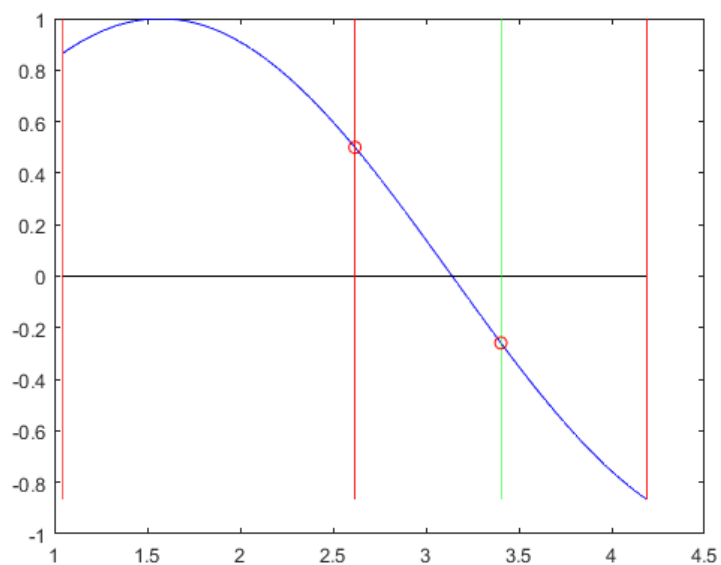
6. Prekriti zelenu vertikalnu osu i tačku iz prethodne iteracije crvenom bojom:

```
plot([zero zero], [fMin fMax], 'red', zero, fZero, 'o red')
```

Ako se prethodni postupak (od tačke 3) ponavlja:

nakon 1. ponavljanja:

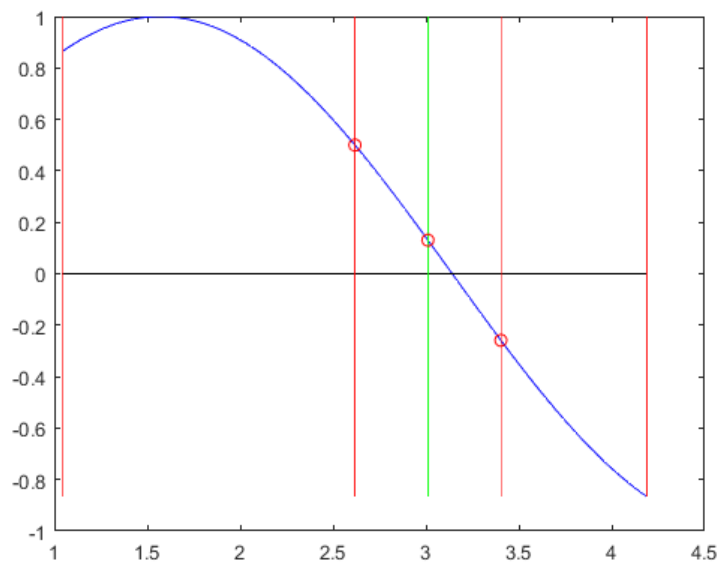
```
fZero =  
    -0.2588
```



Slika 5. Nakon 1. ponavljanja

nakon 2. ponavljanja:

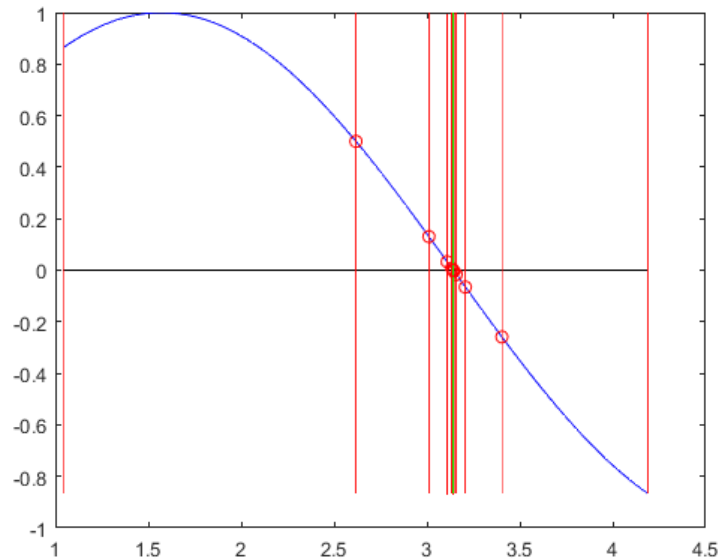
```
fZero =  
    0.1305
```



Slika 6. Nakon 2. ponavljanja

nakon 16. ponavljanja:

```
fZero =  
    7.9895e-06
```



Slika 7. Nakon 16. ponavljanja

7. Prethodni postupak (od tačke 3) se može ugnjeziditi u beskonačnu *while* petlju. Usput je poželjno **brojati iteracije**:

```
it = 1;  
while true  
    zero = (a + b)/2;  
  
    fZero = f(zero);  
    plot([zero zero], [fMin fMax], 'green', zero, fZero, 'o green')  
    ←  
    if f(a)*fZero < 0  
        b = zero;  
    else  
        a = zero;  
    end  
    it = it + 1;  
  
    plot([zero zero], [fMin fMax], 'red', zero, fZero, 'o red')  
end
```

8. Potrebno je definisati uslov za prekid iteracije u slučaju da **vrednost funkcije u pretpostavljenoj nuli padne ispod tražene preciznosti** ili u slučaju da **dužina podintervala padne ispod tražene preciznosti**:

```
if abs(fZero) < errMax || abs(b - a) < errMax  
    return  
end
```

9. Sada je moguće definisati funkciju koja sadrži prethodni algoritam. Na početku funkcije zatvoriti eventualno postojeći prethodni grafik:

```
function [zero, it] = zeroBisection(f, a, b, errMax, plotSpeed)
    close
    .
    .
    .
end
```

10. Pauzirati algoritam u zavisnosti od tražene brzine iscrtavanja postupka:

```
.
.
.

if abs(fZero) < errMax || abs(b - a) < errMax
    return
end
pause(1/plotSpeed)

.
.
.
end
```

11. Testirati funkciju zeroBisection na primeru:

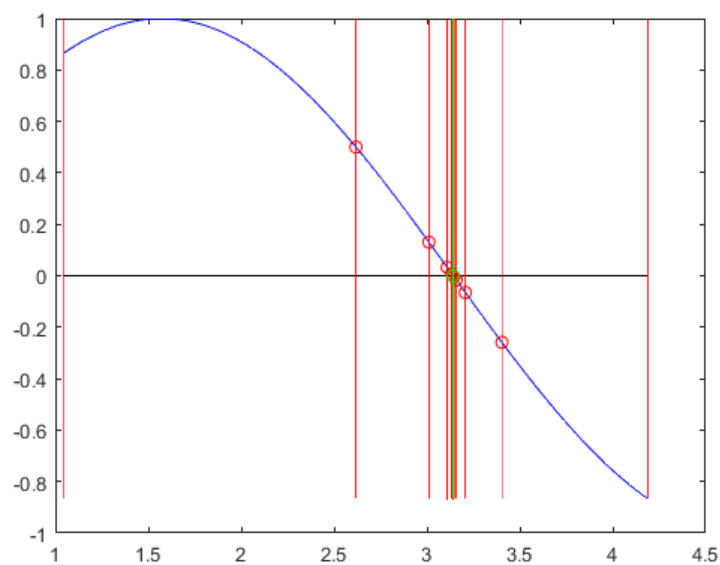
```
f = @(x) sin(x);
a = pi/3;
b = 4*pi/3;

[zero, it] = zeroBisection(f, a, b, 10^-5, 10.0)
```

Rezultat:

```
zero =
    3.1416
```

```
it =
    17
```



Slika 8. Nula funkcije

12. Napraviti 2 podfunkcije u funkciji `zeroBisection`. Ako je prosleđena brzina iscrtavanja postupka 0 ili manja, pozvati varijantu funkcije bez naredbi za iscrtavanje i pauziranje algoritma:

```
function [zero, it] = zeroBisection(f, a, b, errMax, plotSpeed)
    if plotSpeed <= 0
        [zero, it] = zeroBisectionWithoutPlot(f, a, b, errMax);
        return
    end
    [zero, it] = zeroBisectionWithPlot(f, a, b, errMax, plotSpeed);
end
```

13. Testirati funkciju `zeroBisection` na primeru:

```
f = @(x) sin(x);
a = pi/3;
b = 4*pi/3;

xf = linspace(a, b, 100);
yf = f(xf);
plot(xf, yf, 'blue'), hold on
plot([a b], [0 0], 'black')

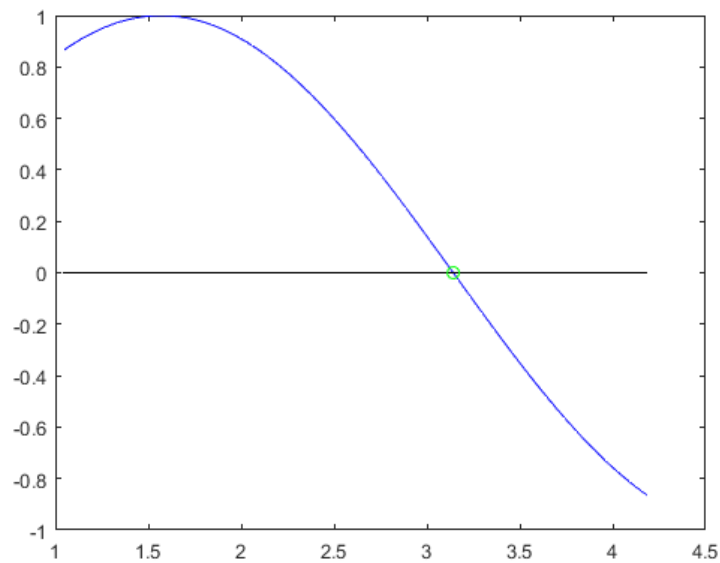
[zero, it] = zeroBisection(f, a, b, 10^-5, 0.0)
fZero = f(zero)
scatter(zero, fZero, 'green'), hold off
```

Rezultat:

```
zero =
    3.1416

it =
    17

fZero =
    7.9895e-06
```



Slika 9. Nula funkcije

2. Metoda sečice

Zadatak 1. Napisati metodu sečice za traženje nule nelinearnih funkcija.

Pokušati prvo ručno jednu iteraciju metode sečice nad funkcijom $f(x) = \sin x$ na intervalu $x \in \left[\frac{\pi}{3}, \frac{4\pi}{3}\right]$:

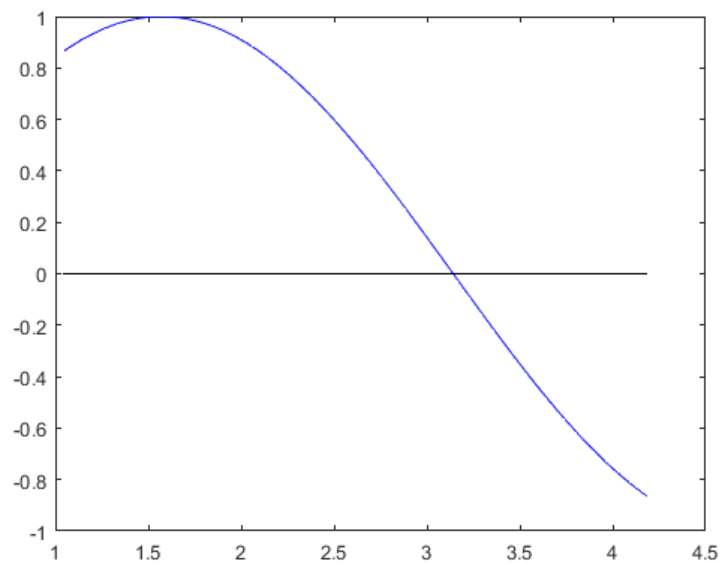
```
f = @(x) sin(x);  
a = pi/3;  
b = 4*pi/3;
```

```
plotA = pi/3;  
plotB = 4*pi/3;
```

1. Nacrtati funkciju nad intervalom i naći njen minimum i maksimum i zadržati grafik:

```
x = linspace(plotA, plotB, 100);  
fX = f(x);  
fMin = min(fX);  
fMax = max(fX);  
plot(x, fX, 'blue', [plotA plotB], [0 0], 'black'), hold on
```

Rezultat:

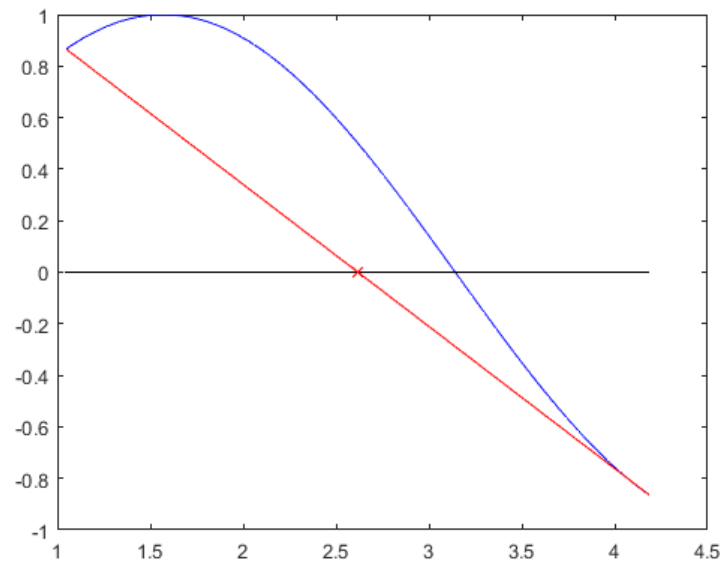


Slika 10. Grafik funkcije

2. Naći nulu sečice. Nacrtati sečicu i nacrtati znak 'x' u njenoj nuli crvenom bojom:

```
fA = f(a);  
fB = f(b);  
zero = b - fB*(b - a)/(fB - fA);  
plot([a b], [fA fB], 'red', zero, 0, 'x red')
```

Rezultat:



Slika 11. Sečica

3. Pretpostaviti nulu funkcije u nuli sečice i izračunati vrednost funkcije u pretpostavljenoj nuli:

```
fZero = f(zero)
```

Rezultat:

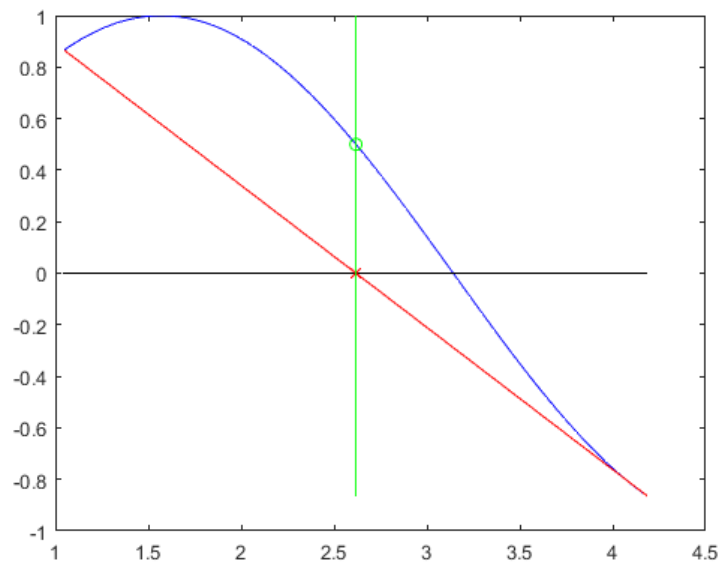
```
fZero =  
    0.5000
```

Izračunata vrednost je **različita od 0**.

4. Nacrtati vertikalnu osu i tačku određenu izračunatom vrednošću u pretpostavljenoj nuli funkcije zelenom bojom:

```
plot([zero zero], [fMin fMax], 'green', zero, fZero, 'o green')
```

Rezultat:



Slika 12. Pretpostavljena nula

5. Pripremiti podinterval za narednu iteraciju. Proglasiti kraj intervala za novi početak, a pretpostavljenu nulu za novi kraj intervala:

```
a = b;  
b = zero;
```

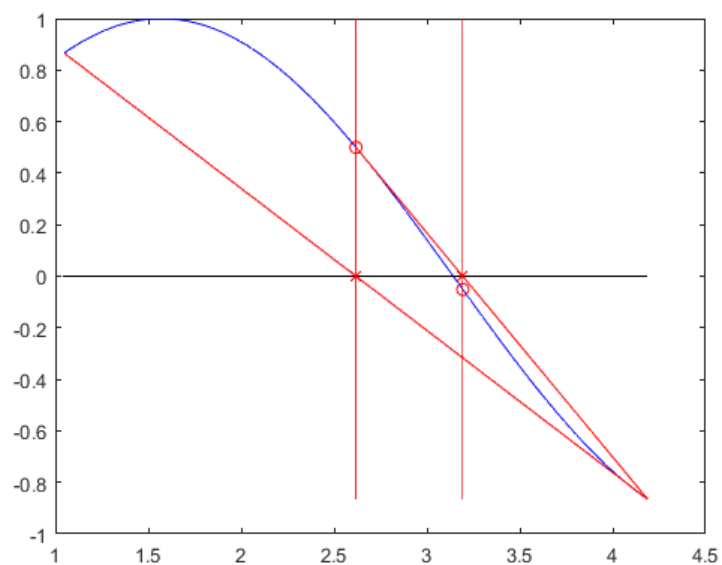
6. Prekriti zelenu vertikalnu osu i tačku iz prethodne iteracije crvenom bojom:

```
plot([zero zero], [fMin, fMax], 'red', zero, fZero, 'o red')
```

Ako se prethodni postupak (od tačke 2) ponavlja:

nakon 1. ponavljanja:

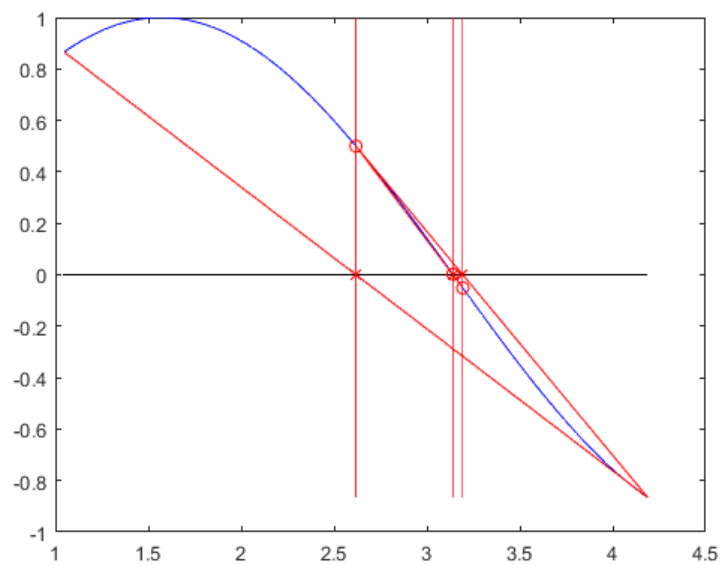
```
fZero =  
-0.0513
```



Slika 13. Nakon 1. ponavljanja

nakon 2. ponavljanja:

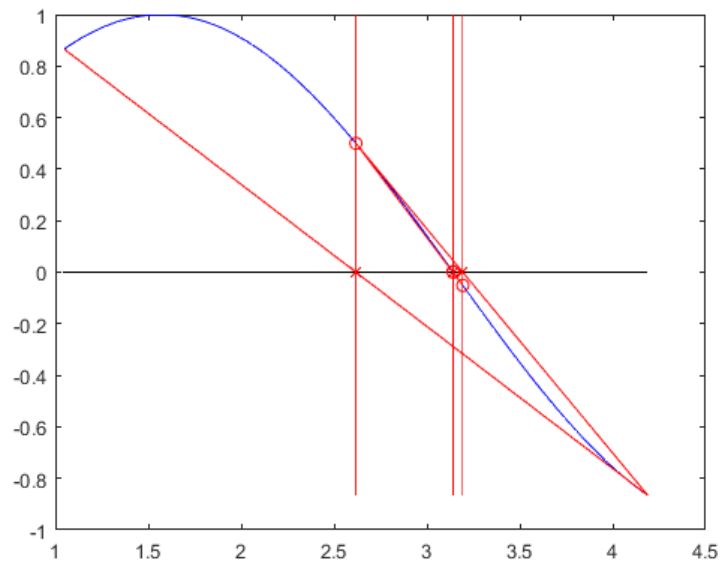
```
fZero =  
0.0022
```



Slika 14. Nakon 2. ponavljanja

nakon 3. ponavljanja:

```
fZero =  
-9.1638e-07
```



Slika 15. Nakon 3. ponavljanja

7. Metoda sečice nema zagarantovanu konvergenciju. Prethodni postupak (od tačke 2) se može ugnjeziditi u *for* petlju sa **ograničenim brojem iteracija**:

```
for it = 1:itMax  
    fA = f(a);  
    fB = f(b);  
    zero = b - fB*(b - a)/(fB - fA);  
    plot([a b], [fA fB], 'red', zero, 0, 'x red')  
    fZero = f(zero);  
    plot([zero zero], [fMin, fMax], 'green', zero, fZero, 'o green')  
  
    a = b;  
    b = zero;  
  
    plot([zero zero], [fMin, fMax], 'red', zero, fZero, 'o red')  
end
```

8. Potrebno je definisati uslov za prekid iteracije u slučaju da **vrednost funkcije u pretpostavljenoj nuli padne ispod tražene preciznosti**:

```
if abs(fZero) < errMax  
    return  
end
```

9. Sada je moguće definisati funkciju koja sadrži prethodni algoritam. Na početku funkcije zatvoriti eventualno postojeći prethodni grafik:

```
function [zero, it] = zeroSecant(f, a, b, errMax, itMax, plotSpeed, plotA, plotB)
    close
    .
    .
    .
end
```

10. Pauzirati algoritam u zavisnosti od tražene brzine iscrtavanja postupka:

```
.
.
.
if abs(fZero) < errMax
    return
end
pause(1/plotSpeed)

.
.
.
end
```

11. Testirati funkciju zeroSecant.m na primeru:

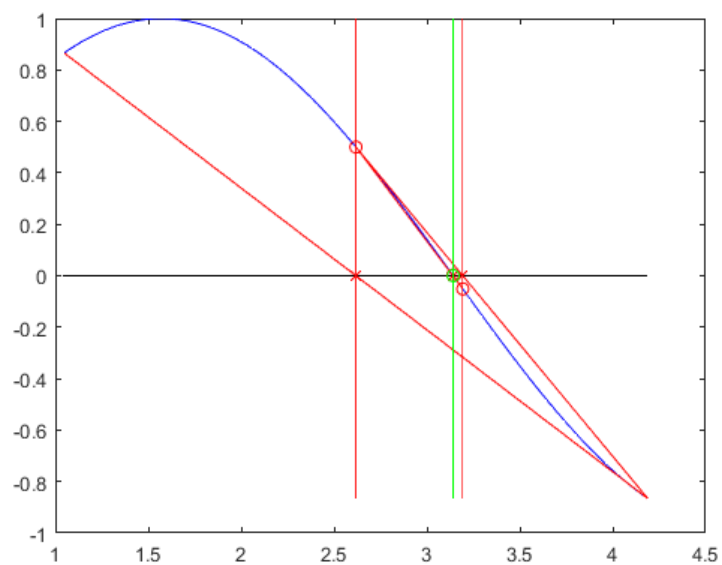
```
f = @(x) sin(x);
a = pi/3;
b = 4*pi/3;

[zero, it] = zeroSecant(f, a, b, 10^-5, 100, 2.0, a, b)
```

Rezultat:

```
zero =
    3.1416
```

```
it =
    4
```



Slika 16. Nula funkcije

12. Napraviti 2 podfunkcije u funkciji `zeroSecant`. Ako je prosleđena brzina iscrtavanja postupka 0 ili manja, pozvati varijantu funkcije bez naredbi za iscrtavanje i pauziranje algoritma:

```
function [zero, it] = zeroSecant(f, a, b, errMax, itMax, plotSpeed, plotA, plotB)
    if plotSpeed <= 0
        [zero, it] = zeroSecantWithoutPlot(f, a, b, errMax, itMax);
        return
    end
    [zero, it] = zeroSecantWithPlot(f, a, b, errMax, itMax, plotSpeed, plotA, plotB);
end
```

13. Testirati funkciju `zeroSecant.m` na primeru:

```
f = @(x) sin(x);
a = pi/3;
b = 4*pi/3;

xf = linspace(a, b, 100);
yf = f(xf);
plot(xf, yf, 'blue'), hold on
plot([a b], [0 0], 'black')

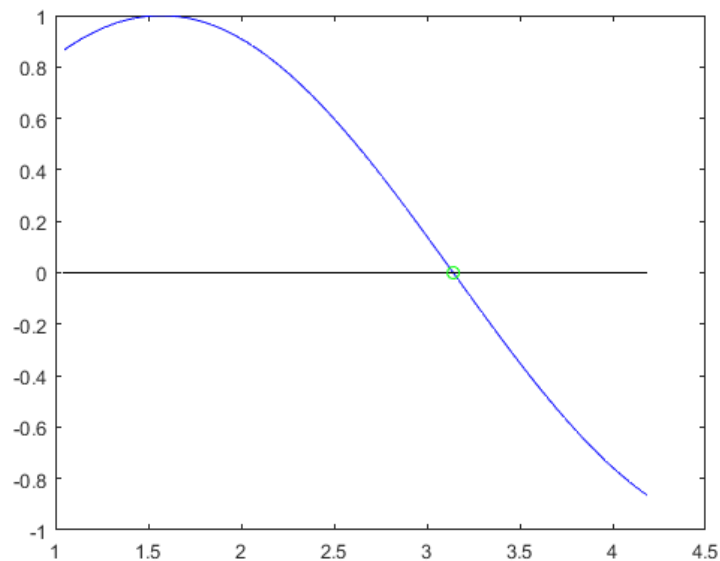
[zero, it] = zeroSecant(f, a, b, 10^-5, 100, 0.0)
fZero = f(zero)
scatter(zero, fZero, 'green'), hold off
```

Rezultat:

```
zero =
    3.1416
```

```
it =
    4
```

```
fZero =
   -9.1638e-07
```

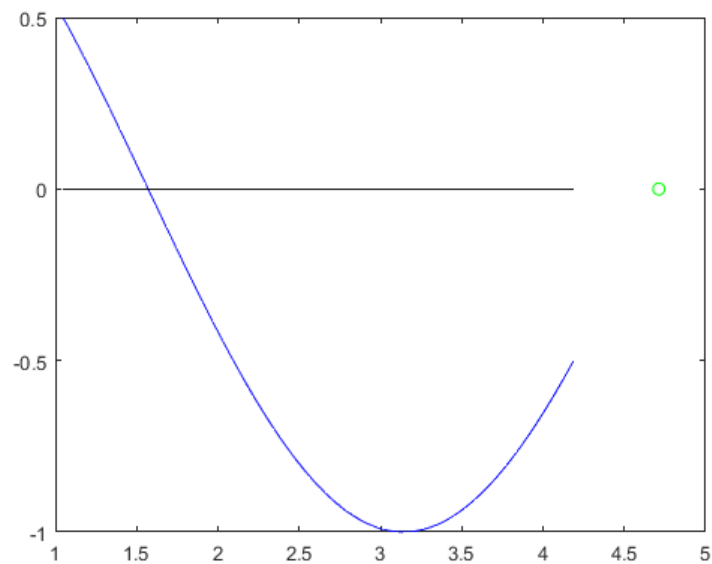


Slika 17. Nula funkcije

14. Testirati funkciju `zeroSecant.m` za funkciju $f(x) = \cos x$ na intervalu $x \in \left[\frac{\pi}{3}, \frac{4\pi}{3}\right]$.

Rezultat:

```
zero =  
    4.7124  
  
it =  
    6  
  
fZero =  
    8.6152e-06
```



Slika 18. Nula funkcije van intervala

Metoda sečice je **otvorena metoda**. Pronađena **nula funkcije je van traženog intervala**. Problem se može rešiti odabrom drugačijeg početnog intervala ili upotrebom zatvorene metode.

Ako je argument `plotSpeed` veći od 0 u ovom slučaju radi pravilnog iscrtavanja postupka, potrebno je proširiti interval iscrtavanja argumentima `plotA` i `plotB`.

* **Zadatak 3.** Napisati metodu *False Position* za traženje nule nelinearnih funkcija.

3. Njutnova metoda

Zadatak 4. Napisati Njutnovu metodu za traženje nule nelinearnih funkcija.

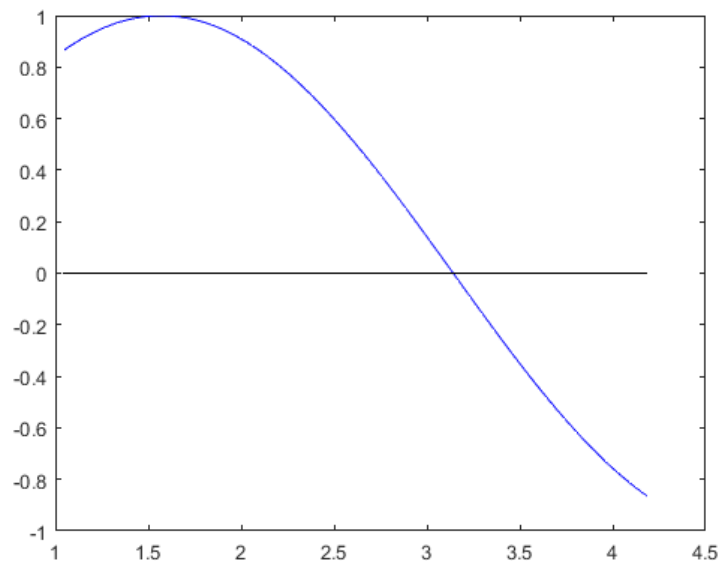
Pokušati prvo ručno jednu iteraciju Njutnove metode nad funkcijom $f(x) = \sin x$. Analitički izvod funkcije je $f'(x) = \cos x$. Početi sa kraja intervala $x_0 = \frac{4\pi}{3}$:

```
f = @(x) sin(x);  
%df = matlabFunction(diff(sym(f))); % simboličko određivanje izvoda; mora biti instaliran  
Symbolic Math Toolbox  
df = @(x) cos(x);  
x0 = 4*pi/3;  
  
plotA = pi/3;  
plotB = 4*pi/3;
```

1. Nacrtati funkciju nad intervalom i naći njen minimum i maksimum i zadržati grafik:

```
x = linspace(plotA, plotB, 100);  
fX = f(x);  
fMin = min(fX);  
fMax = max(fX);  
plot(x, fX, 'blue', [plotA plotB], [0 0], 'black'), hold on
```

Rezultat:

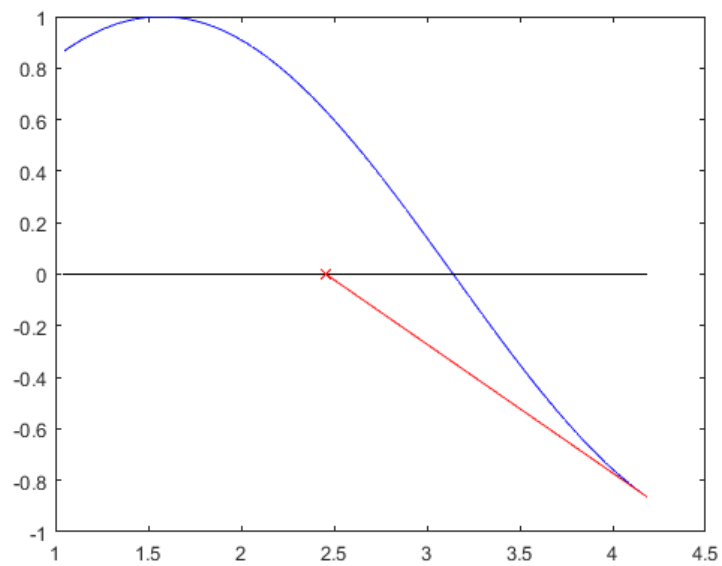


Slika 19. Grafik funkcije

2. Naći nulu tangente. Nacrtati tangentu i nacrtati znak 'x' u njenoj nuli crvenom bojom:

```
zero = x0 - f(x0)/df(x0);  
plot([x0 zero], [f(x0) 0], 'red', zero, 0, 'x red')
```

Rezultat:



Slika 20. Tangenta

3. Pretpostaviti nulu funkcije u nuli tangente i izračunati vrednost funkcije u pretpostavljenoj nuli:

```
fZero = f(zero)
```

Rezultat:

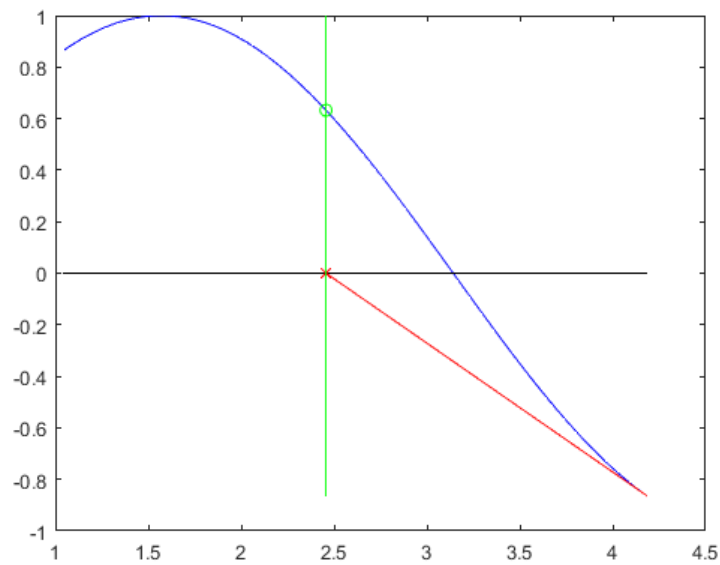
```
fZero =  
    0.6326
```

Izračunata vrednost je **različita od 0**.

4. Nacrtati vertikalnu osu i tačku određenu izračunatom vrednošću u pretpostavljenoj nuli funkcije zelenom bojom:

```
plot([zero zero], [fMin fMax], 'green', zero, fZero, 'o green')
```

Rezultat:



Slika 21. Pretpostavljena nula

5. Pripremiti podinterval za narednu iteraciju. Proglasiti pretpostavljenu nulu za novu početnu tačku:

```
x0 = zero;
```

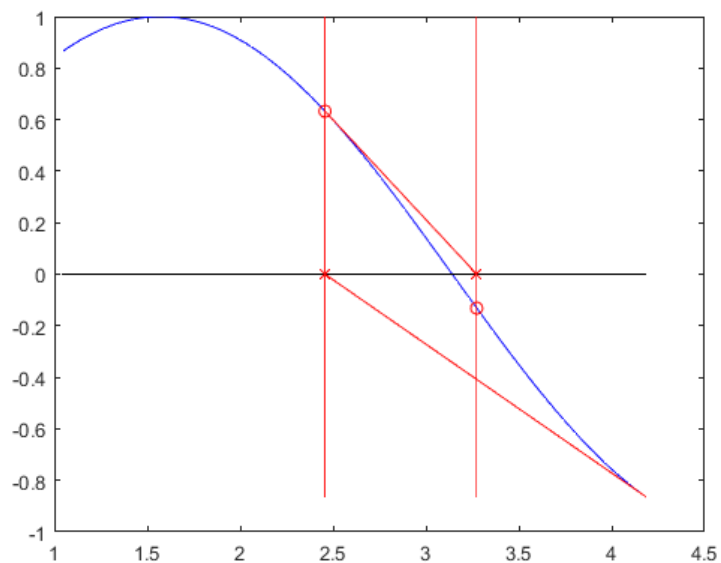
6. Prekriti zelenu vertikalnu osu i tačku iz prethodne iteracije crvenom bojom:

```
plot([zero zero], [fMin, fMax], 'red', zero, fZero, 'o red')
```

Ako se prethodni postupak (od tačke 2) ponavlja:

nakon 1. ponavljanja:

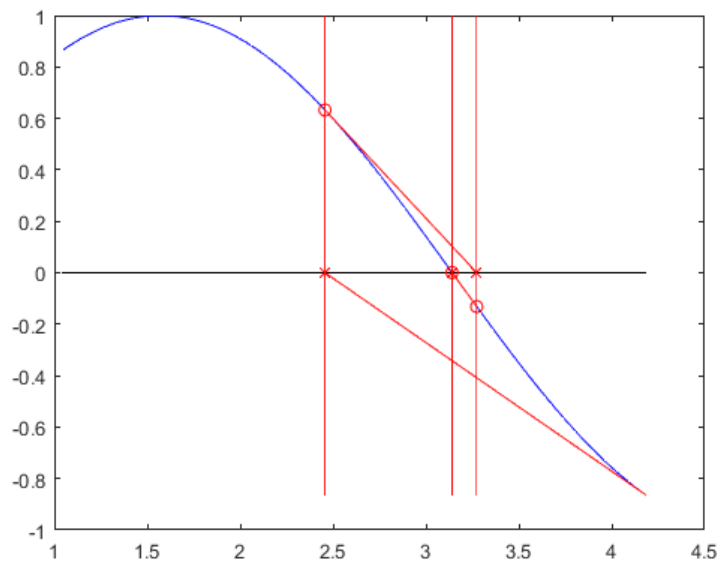
```
fZero =  
-0.1315
```



Slika 22. Nakon 1. ponavljanja

nakon 2. ponavljanja:

```
fZero =  
7.6969e-04
```



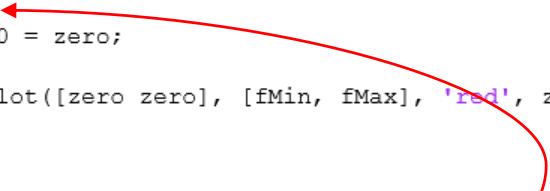
Slika 23. Nakon 2. ponavljanja

nakon 3. ponavljanja:

```
fZero =  
    -1.5199e-10
```

7. Njutnova metoda nema zagarantovanu konvergenciju. Prethodni postupak (od tačke 2) se može ugnjeziditi u *for* petlju sa **ograničenim brojem iteracija**:

```
for it = 1:itMax  
    zero = x0 - f(x0)/df(x0);  
    plot([x0 zero], [f(x0) 0], 'red', zero, 0, 'x red')  
  
    fZero = f(zero);  
    plot([zero zero], [fMin, fMax], 'green', zero, fZero, 'o green')  
    x0 = zero;  
    plot([zero zero], [fMin, fMax], 'red', zero, fZero, 'o red')  
end
```



8. Potrebno je definisati uslov za prekid iteracije u slučaju da **vrednost funkcije u pretpostavljenoj nuli padne ispod tražene preciznosti**:

```
if abs(fZero) < errMax  
    return  
end
```

9. Sada je moguće definisati funkciju koja sadrži prethodni algoritam. Na početku funkcije zatvoriti eventualno postojeći prethodni grafik:

```
function [zero, it] = zeroNewton(f, df, x0, errMax, itMax, plotSpeed, plotA, plotB)  
    close  
    .  
    .  
    .  
end
```

10. Pauzirati algoritam u zavisnosti od tražene **brzine iscrtavanja postupka**:

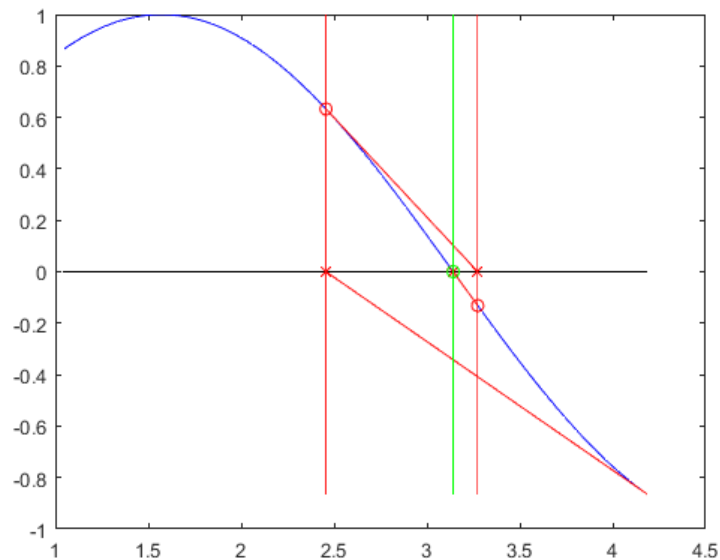
```
.  
.   
.   
  
if abs(fZero) < errMax  
    return  
end  
pause(1/plotSpeed)  
  
.   
.   
.   
end
```

11. Testirati funkciju zeroNewton.m na primeru:

```
f = @(x) sin(x);  
df = @(x) cos(x);  
x0 = 4*pi/3;  
  
[zero, it] = zeroNewton(f, df, x0, 10^-5, 100, 2.0, pi/3, 4*pi/3)
```

Rezultat:

```
zero =  
    3.1416  
  
it =  
    4
```



Slika 24. Nula funkcije

12. Napraviti 2 podfunkcije u funkciji zeroNewton. Ako je prosleđena brzina iscrtavanja postupka 0 ili manja, pozvati varijantu funkcije bez naredbi za iscrtavanje i pauziranje algoritma:

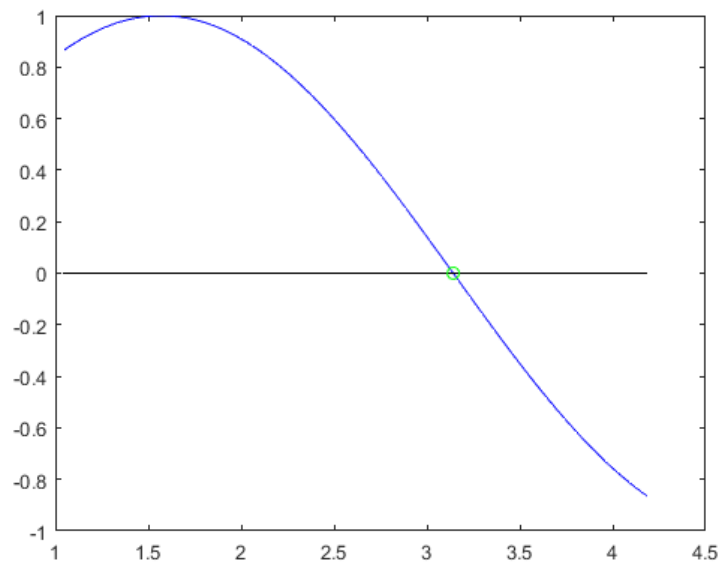
```
function [zero, it] = zeroNewton(f, df, x0, errMax, itMax, plotSpeed, plotA, plotB)  
    if plotSpeed <= 0  
        [zero, it] = zeroNewtonWithoutPlot(f, df, x0, errMax, itMax);  
        return  
    end  
    [zero, it] = zeroNewtonWithPlot(f, df, x0, errMax, itMax, plotSpeed, plotA,  
    plotB);  
end
```

13. Testirati funkciju `zeroNewton.m` na primeru:

```
f = @(x) sin(x);  
df = @(x) cos(x);  
x0 = 4*pi/3;  
  
xf = linspace(pi/3, 4*pi/3, 100);  
yf = f(xf);  
plot(xf, yf, 'blue'), hold on  
plot([a b], [0 0], 'black')  
  
[zero, it] = zeroNewton(f, df, x0, 10^-5, 100, 0.0)  
fZero = f(zero)  
scatter(zero, fZero, 'green'), hold off
```

Rezultat:

```
zero =  
    3.1416  
  
it =  
    4  
  
fZero =  
-1.5199e-10
```



Slika 25. Nula funkcije