REŠAVANJE NELINEARNIH JEDNAČINA ITERATIVNE METODE

predavač:
Aleksandar Kovačević

Rešavanje nelinearnih jednačina

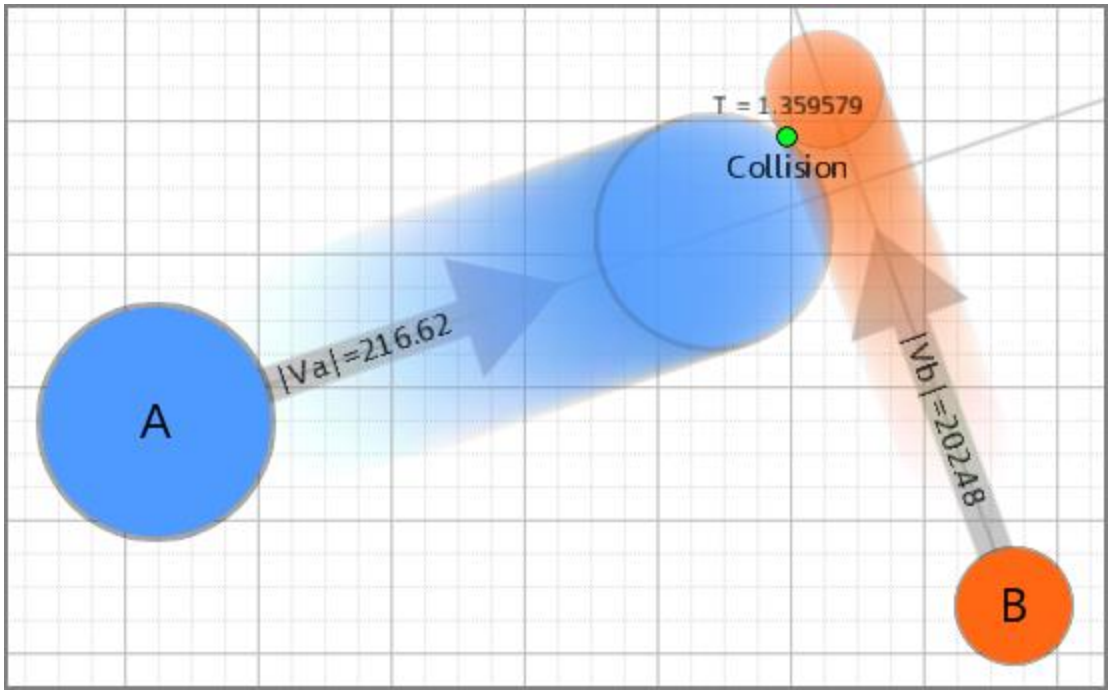
- Na prethodna dva predavanja bavili smo se linearnim jednačinama.
- Međutim, modeli realnog sveta nisu uvek linearni, često moramo da koristimo nelinearne funkcije (nrp. polinome velikog stepena, e^x, sin(x), cos(x) itd).
- Rešavanje jednačina sa ovakvim fukcijama analitičkim putem je često veoma teško.

Motivacioni primer
Detekcija kolizija u video igrama

- Kako detektovati da li i kad će doći do kolizije dva objekta u video igrama?

Detekcija kolizija
linearno kretanje

- Prvo ćemo posmatrati linearno kretanje.
- Zadatak nam je da odredimo da li i kad će doći do kolizije dve sfere koje se kreću linearno.



Detekcija kolizija
linearno kretanje

- Ako sfere A i B kreću iz pozcija P_a i P_b i imaju konstantne brzine v_a i v_b, njihovo kretanje može se opisati sledećim jednačinama:

$$A(t) = P_a + tV_a$$
$$B(t) = P_b + tV_b$$

Detekcija kolizija
linearno kretanje

- Definišemo funkciju udaljenosti dve sfere po vremenu.

$$d(t) = |A(t) - B(t)| - (R_a + R_b)$$

- gde su A(t) i B(t) pozicije centara sfera, a R_a i R_b njihovi poluprečnici.
- Koristimo Euklidsko rastojanje za: $|A(t) - B(t)|$
- Tako dobijamo

$$d(t) = \sqrt{(A(t) - B(t))^2} - (R_a + R_b)$$

Detekcija kolizija
linearno kretanje

- Do kolizije dolazi u trenutku t za koji d(t)=0.
- Kako odrediti da li postoji takvo t?
- Rešavamo kvadratnu jednačinu:

$$0 = \sqrt{\left(A(t) - B(t)\right)^2} - (R_a + R_b)$$

Detekcija kolizija
linearno kretanje

- Analitičko rešenje:

$$(R_a + R_b)^2 = (P_a + tV_a - P_b - tV_b)^2$$

$$(R_a + R_b)^2 = \big((P_a - P_b) + t(V_a - V_b)\big)^2$$

$$P_{ab} = P_a - P_b \qquad V_{ab} = V_a - V_b$$

- ....

$$(R_a + R_b)^2 = (P_{ab} + tV_{ab})^2$$

$$0 = t^2(V_{ab} \cdot V_{ab}) + 2t(P_{ab} \cdot V_{ab}) + (P_{ab} \cdot P_{ab}) - (R_a + R_b)^2$$

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

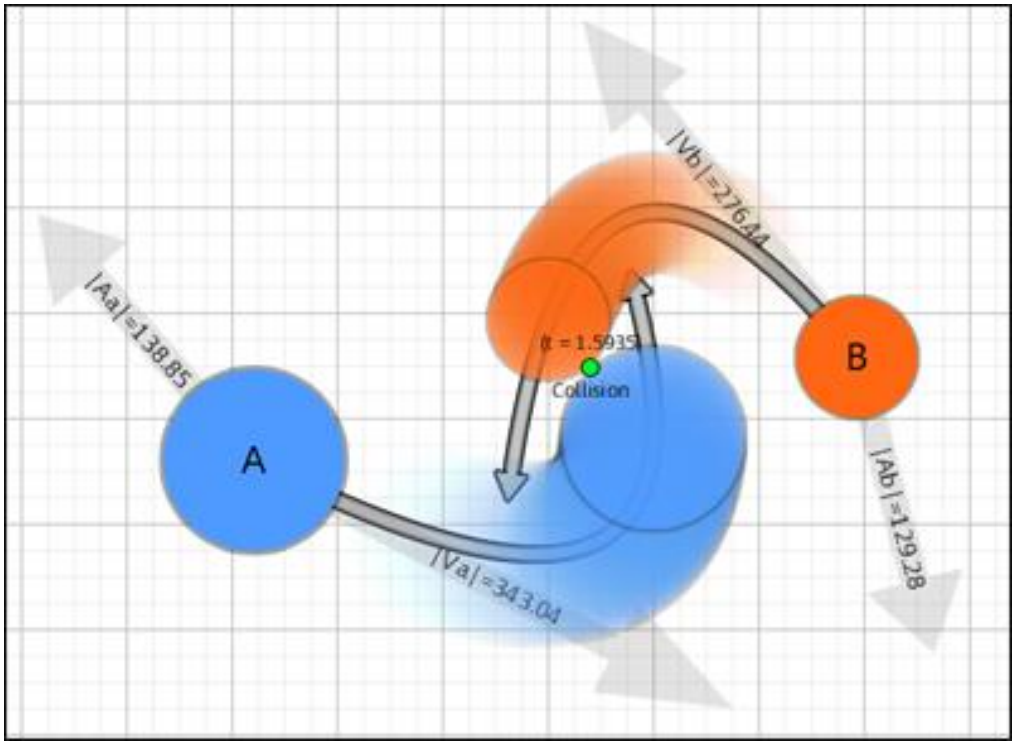
$$a = V_{ab} \cdot V_{ab} \qquad \text{b}=2(P_{ab} \cdot V_{ab}) \qquad \text{c}=P_{ab} \cdot P_{ab} - (R_a + R_b)^2$$

Detekcija kolizija
linearno kretanje

- Moguća su 3 slučaja:
  - nema realnih nula – sfere se nikad nisu niti će se sudariti.
  - jedna realna nula – sfere će dodirnuti (okrznuti) jedna drugu u jednoj tački.
  - dve realne nule – sfere će se sudariti tj. proći će jedna kroz drugu u nekom trenutku (taj trenutak je nula koja ima manju vrednost).

Detekcija kolizija
nelinearno kretanje

- Prethodni primer bilo je lako rešiti analitički, međutim kretanje objekata njie uvek linearno.
- Šta ako imamo dve sfere koje se kreću po paraboli?



Detekcija kolizija
nelinearno kretanje

- Jednačine kretanja sfera sad imaju i ubrzanje A_a (A_b):

$$P_a(t) = P_a + tV_a + t^2 A_a$$

$$P_b(t) = P_b + tV_b + t^2 A_b$$

Detekcija kolizija
nelinearno kretanje

- Opet rešavamo nelinearnu jednačinu za udaljenost:

$$0 = \sqrt{\left(A(t) - B(t)\right)^2} - (R_a + R_b)$$

- Sada su jednačine za A(t) i B(t) komplikovanije.

Detekcija kolizija nelinearno kretanje

- Analitičko rešenje:

$$P_{ab} = P_a - P_b$$

$$V_{ab} = V_a - V_b$$

$$A_{ab} = A_a - A_b$$

$$R_{ab} = R_a + R_b$$

$$d(t) = |P_{ab} + tV_{ab} + t^2 A_{ab}| - R_{ab}$$

$$0 = (P_{ab} \cdot P_{ab}) + 2t(P_{ab} \cdot V_{ab}) + 2t^2((P_{ab} \cdot A_{ab}) + (V_{ab} \cdot V_{ab})) + 2t^3(V_{ab} \cdot A_{ab}) + t^4(A_{ab} \cdot A_{ab}) - R_{ab}^2$$

Detekcija kolizija
nelinearno kretanje

- Treba naći nule ovog polinoma:

$$t^4 a + t^3 b + t^2 c + td + e = 0$$

$$a = (A_{ab} \cdot A_{ab})$$

$$b = 2(V_{ab} \cdot A_{ab})$$

$$c = 2\big((P_{ab} \cdot A_{ab}) + (V_{ab} \cdot V_{ab})\big)$$

$$d = 2(P_{ab} \cdot V_{ab})$$

$$e = (P_{ab} \cdot P_{ab}) - R_{ab}^2$$

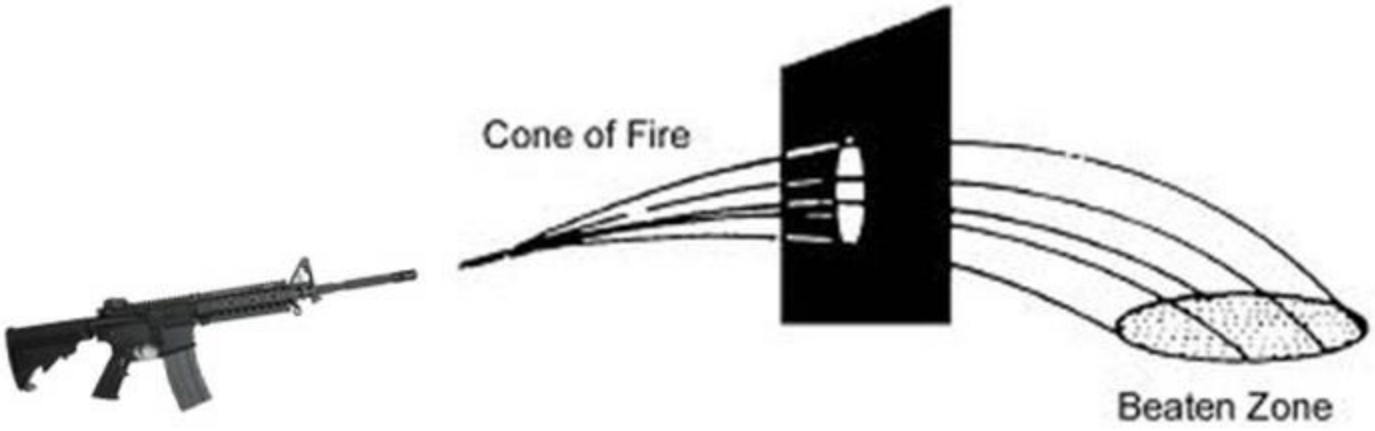
Detekcija kolizija
nelinearno kretanje

- Pomoću softvera *Mathematica* imamo "jednostavno" analitičko rešenje:

```
t = -(b/(4 a))-1/2 Sqrt(b^2/(4 a^2)-(2 c)/(3 a)+(2^(1/3) (c^2-3 b d+12 a e))/(3 a (2 c^3-9 b c d+27 a
    d^2+27 b^2 e-72 a c e+Sqrt(-4 (c^2-3 b d+12 a e)^3+(2 c^3-9 b c d+27 a d^2+27 b^2 e-72
    a c e)^2))^(1/3))+(1/(3 2^(1/3) a))((2 c^3-9 b c d+27 a d^2+27 b^2 e-72 a c e+Sqrt(-4 (c^
      2-3 b d+12 a e)^3+(2 c^3-9 b c d+27 a d^2+27 b^2 e-72 a c e)^2))^(1/3)))-1/2 Sqrt(b
        ^2/(2 a^2)-(4 c)/(3 a)-(2^(1/3) (c^2-3 b d+12 a e))/(3 a (2 c^3-9 b c d+27 a d^2+27 b^2 e-72 a c e+Sqrt(
        -4 (c^2-3 b d+12 a e)^3+(2 c^3-9 b c d+27 a d^2+27 b^2 e-72 a c e)^2))^(1/3))-(1/(3 2^(1/3) a))((2 c
        ^3-9 b c d+27 a d^2+27 b^2 e-72 a c e+Sqrt(-4 (c^2-3 b d+12 a e)^3+(2 c^3-9 b c d+27 a d^2+
        27 b^2 e-72 a c e)^2))^(1/3))-(-(b^3/a^3)+(4 b c)/a^2-(8 d)/a)/(4 Sqrt(b^2/(4 a^2)-(2 c)/(3 a)+(2^(1/3) (c^2-3 b d+12
        a e))/(3 a (2 c^3-9 b c d+27 a d^2+27 b^2 e-72 a c e+Sqrt(-4 (c^2-3 b d+12 a e)^3+(2 c^3-9 b c d+27 a d^2+
        27 b^2 e-72 a c e)^2))^(1/3))+(1/(3 2^(1/3) a))((2 c^3-9 b c d+27 a d^2+27 b^2 e-72 a c e+Sqrt(-4 (
        c^2-3 b d+ 12 a e)^3+(2 c^3-9 b c d+27 a d^2+27 b^2 e-72 a c e)^2))^(1/3)))))
```

Detekcija kolizija
nelinearno kretanje

- Prethodni primer sveo se na polinom četvrtog stepena, a šta bi recimo bilo sa ispaljivanjem projektila u video igri?



- Tu već imamo kosi hitac tj. sinuse, kosinuse itd.

Detekcija kolizija
nelinearno kretanje

- Prethodni primeri ilustruju potrebu za alternativnim metodama za rešavanja nelinearnih jednačina.
- Dobru alternativu predstavljaju numeričke metode.
- Na današnjem predavanju pokazaćemo četiri najpoznatije metode.

Rešavanje nelinearnih jednačina

- Primeri nelinearnih jednačina:

$$\cos(x)=x \text{ ili } e^{x}-16x^{4}+3\sin(x)=17$$

- Iz primera možemo primetiti da:
  - se rešavanje nelinearne jednačine uvek može svesti na određivanje nule nelinearne funkcije.
  - npr. $\cos(x)-x=0$ ili $e^{x}-16x^{4}+3\sin(x)-17=0$
- Metode koje ćemo danas učiti koriste se za pronalaženje nula proizvoljnih funkcija.

Rešavanje nelinearnih jednačina

- Prikazaćemo četiri metode za određivanje nula funkcija.
- U prezentaciji će ravnopravno biti korišćeni termini “nula funkcije” i “rešenje nelinerane jednačine”.
- Ovi termini se (kao što smo videli) odnose na isti pojam.

Iterativni metod (podsećanje)

- Kod iterativnih metoda krećemo od odabrane početne vrednosti x^0
  - bira je korisnik metode (na slučajan ili neki drugi način)
- Koristimo iterativnu formulu koja daje vezu između x_k i x_{k-1}.
- Na taj način izračunavamo niz
  - $x^0, x^1, x^2, ..., x^{k-1}, x^k, ...$

Metode za rešavanje nelinearnih jednačina

Postoji više načina za rešavanje nelinearnih jednačina:

- Analitički metod
  - moguće samo za određene vrste jednačina
- Grafički metod
  - Korisno za prvu procenu lokacije rešenja.
  - Ograničeni smo brojem dimenzija.
- Numeričke metode
  - Otvorene metode
  - Zatvorene metode

Analitički metod

- Moguće je rešiti samo određene tipove jednačina (npr. kvadratna jednačina).

$$a\,x^2 + b\,x + c = 0$$

$$roots = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Ne postoji analitičko rešenje za:

$$x - e^{-x} = 0$$

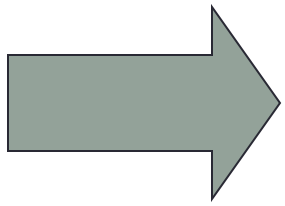
Grafički metod

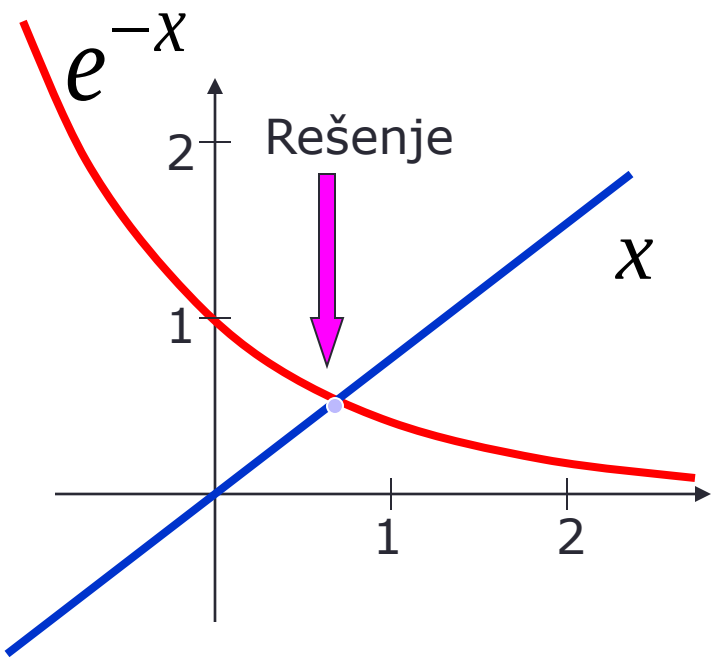
- Dobra osnova za prvu procenu lokacije rešenja i početnog rešenja za numeričku metodu
- Ograničeni smo na 3d.

$$x = e^{-x}$$

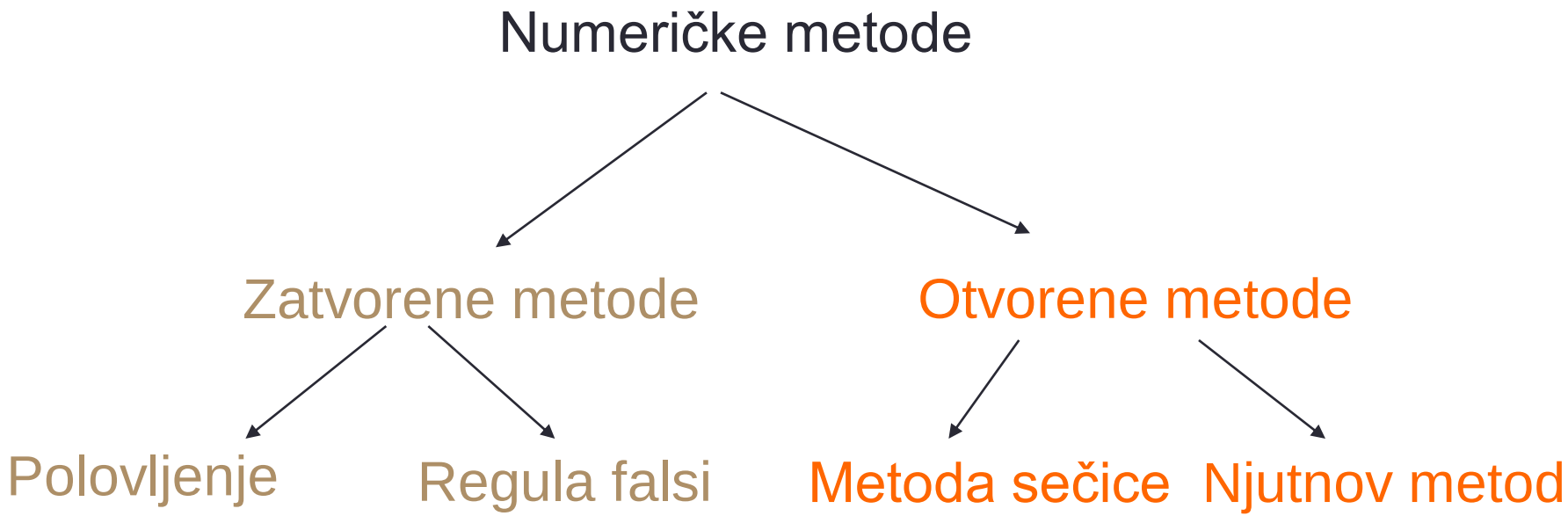
$$x \in [0,1]$$

$$x \approx 0.6$$





Numeričke metode za rešavanje nelinearnih jednačina



Zatvorene metode

- Kod zatvorenih metoda algoritam kreće od zatvorenog intervala koji sadrži rešenje.
- Svakim sledećim korakom taj interval se smanjuje:
- dok se interval ne svede na jednu tačku tj. rešenje ili
- dok veličina intervala ne padne ispod zadate tolerancije (kriterijum koji se koristi u praksi).

Otvorene metode

- Kod otvorenih metoda algoritam kreće od početnog (inicijalnog) rešenja x^0.
- Svakim sledećim korakom dobija se nova procena rešenja $x^0, x^1, x^2, ..., x^{k-1}, x^k, ...$
- Algoritam se zaustavlja kad:
- pronađemo rešenje (tačku za koju važi $f(x)=0$)
- ili

$$|x_k - x_{k-1}| < tolerancija$$

- ili

$$|f(x_k)| < tolerancija$$

Metoda polovljenja

- Jedna od najjednostavnijh metoda za rešavanje nelinearnih jednačina.
- Kao uslov za korišćenje zahteva zatvoreni interval za koji je poznato da sadrži rešenje.

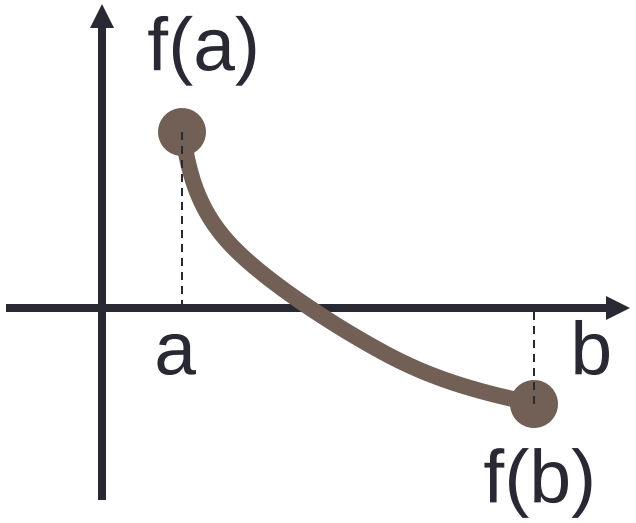
- Metoda polovljenja sistematski smanjuje (polovi) zatvoreni interval.
- Pre svakog polovljenja izvršava se jednostavna provera na osnovu koje se donosi odluka koja polovina se dalje polovi.
- Polovljenje prestaje kad je trenutni interval dovoljno mali.

Teorema srednje vrednosti

- Imamo funkciju f(x) definisanu na intervalu [a,b],

- Teorema srednje vrednosti:

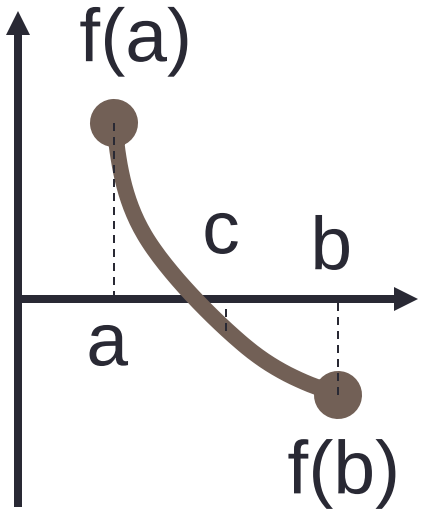
  Ako je f neprekidna i ako su f(a) i f(b) različitog znaka onda funkcija f ima bar jednu nulu na intervalu [a,b].



Metoda polovljenja

Pretpostavke:

- f(x) je neprekidna na [a,b]
- f(a) f(b) < 0



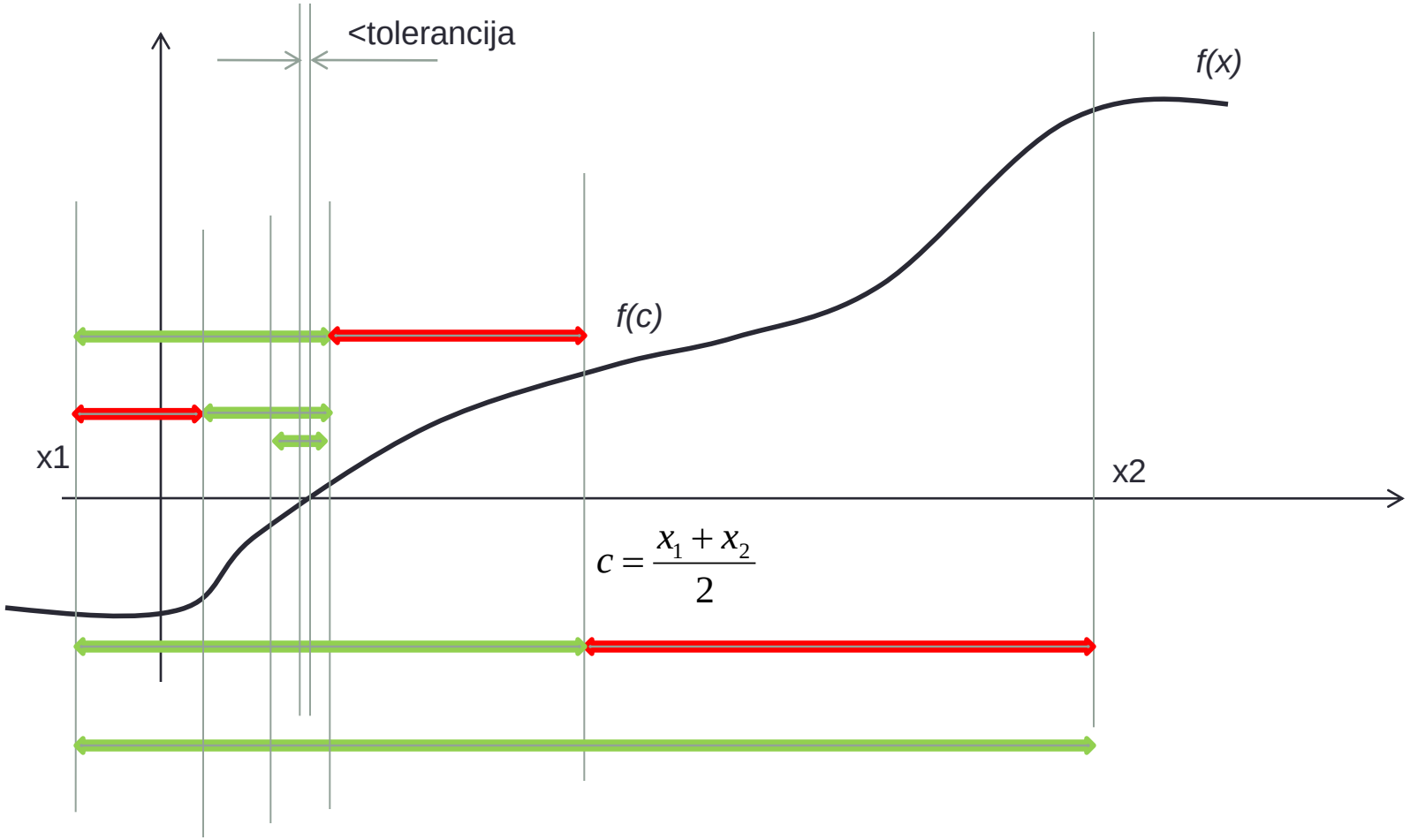
Algoritam:

Loop

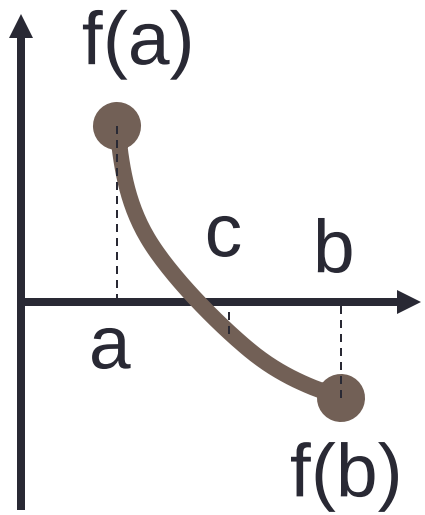
1. Izračunati polovinu [a,b] c=(a+b)/2
2. Izračunati f(c)
3. Ako f(a) f(c) < 0 novi interval je [a, c]
   Ako f(a) f(c) > 0 novi interval je [c, b]
4. Ako $|b-a| < tolerancija$ vrati c=(a+b)/2 kao rešenje

End loop

Metoda polovljenja



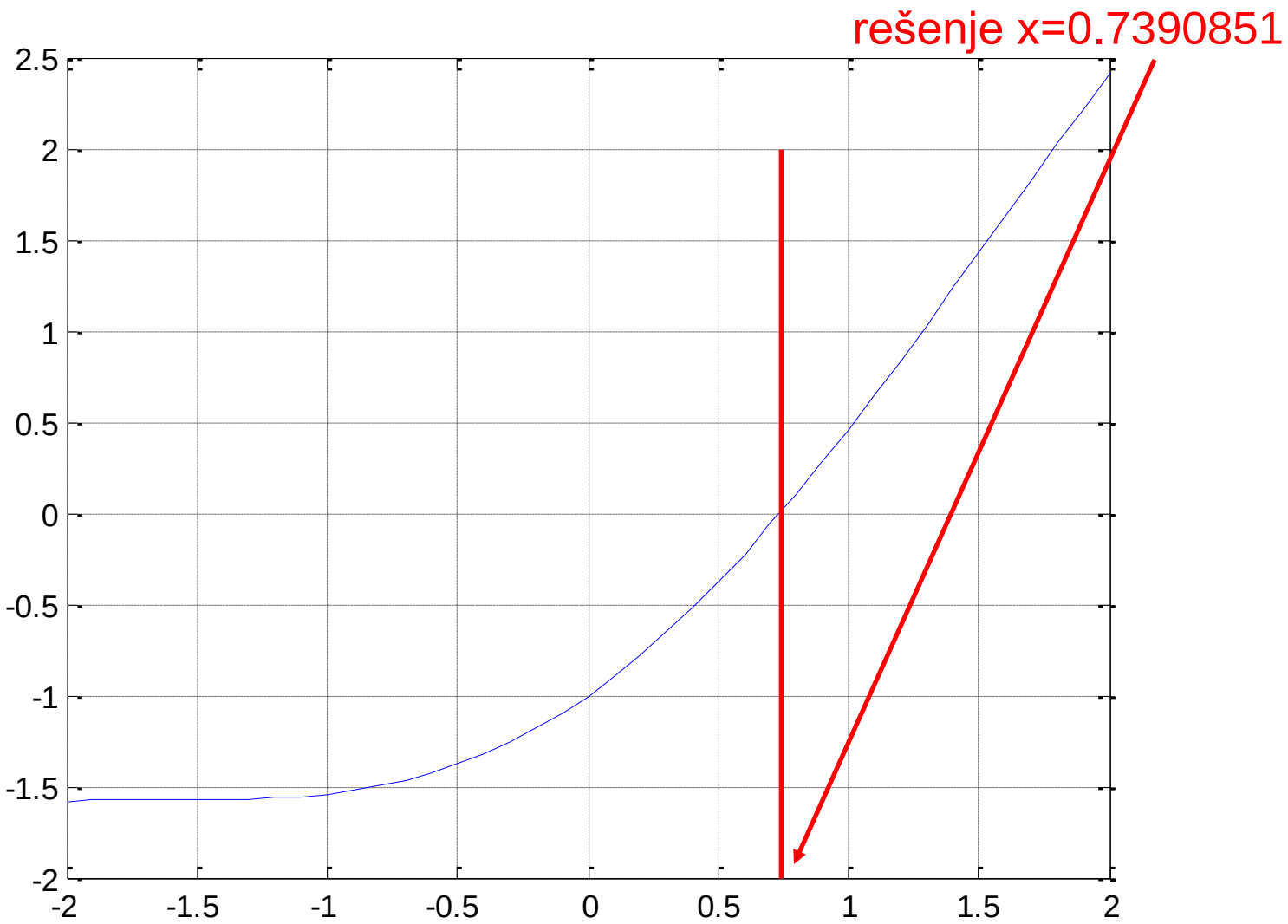
Matlab kod




```
function x=polovljenje(a,b,tacnost,funkcija)
if(feval(funkcija,a)*feval(funkcija,b)<0)
    kraj = 0;
    while(~kraj)
        c=(a+b)/2;
        fc=feval(funkcija,c);
        if(fc==0|abs(b-a)<tacnost)
            kraj=1;
        else
            if(feval(funkcija,c)*feval(funkcija,a)<0)
                b=c;
            else
                a=c;
            end
        end
    end
    x=c;
end
```

feval – izračunava vrednost funkcije “funkcija” u tački a tj. f(a)

Primer
x-cos(x)=0 - koristimo interval [-1,1]



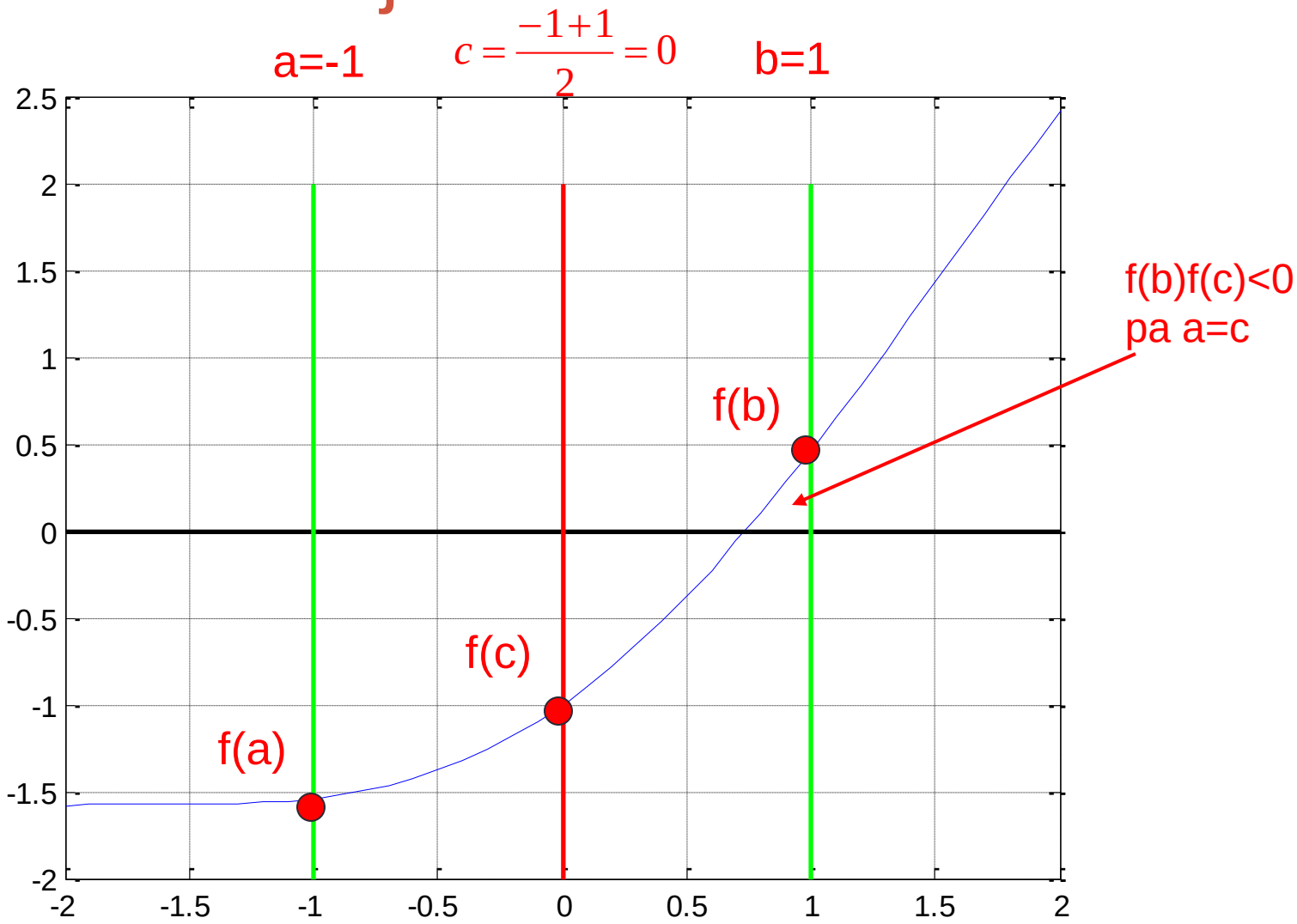
Matlab kod

- Ostaje da objasnimo kako u kodu specificiramo jednačinu koju rešavamo tj. funkciju čiju nulu tražimo.
- Dva od mogućih načina:
  - kreiramo posebnu Matlab funkciju

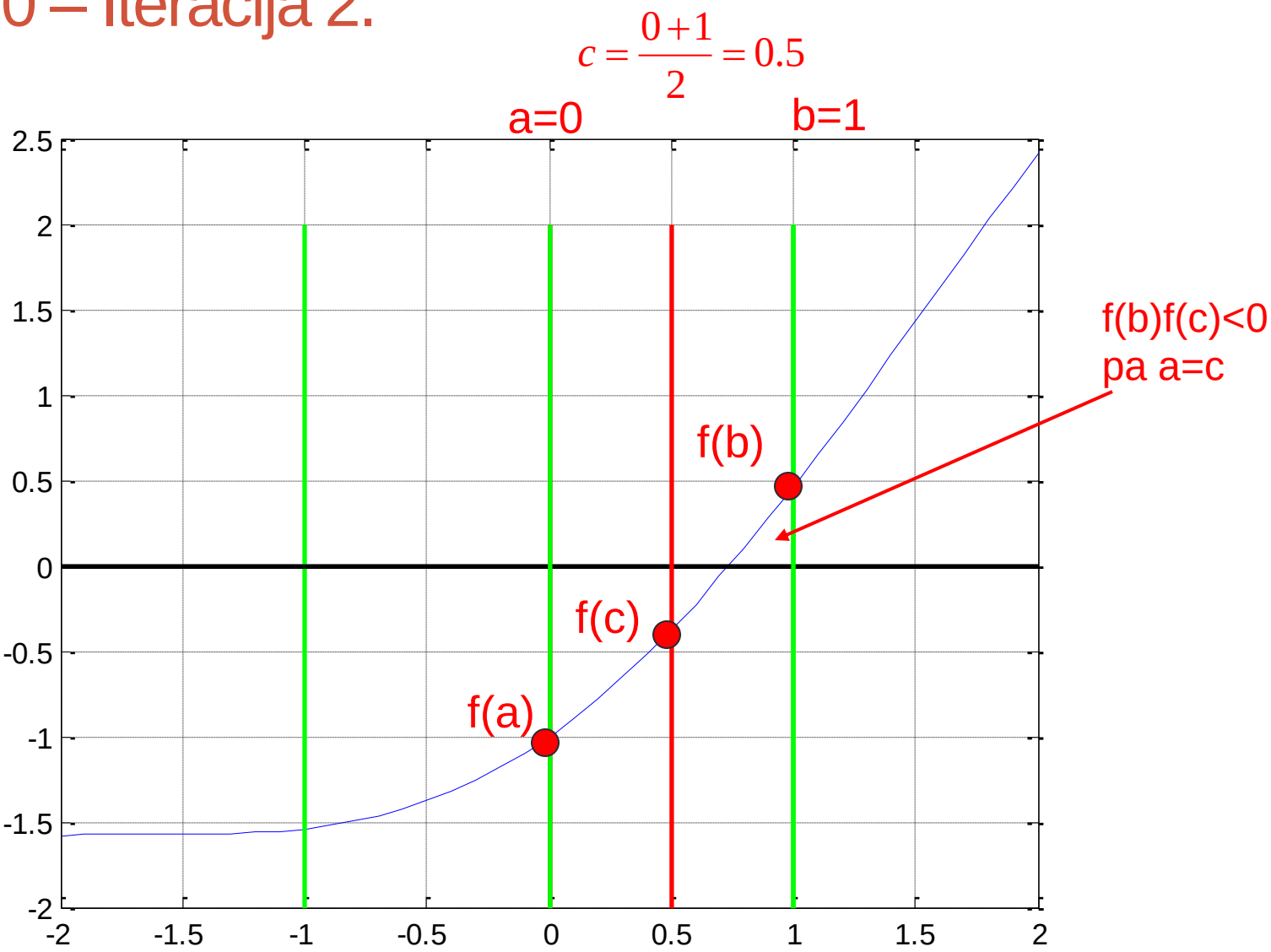
```
function y = jednacina(x)
y=x-cos(x);
```

  - `polovljenje(-1,1,10^-5,'jednacina')`
  - prosleđujemo funkciju direktno u pozivu metode polovljenja
  - `polovljenje(-1,1,10^-5,@(x)x-cos(x))`

Primer
x-cos(x)=0 – Iteracija 1.

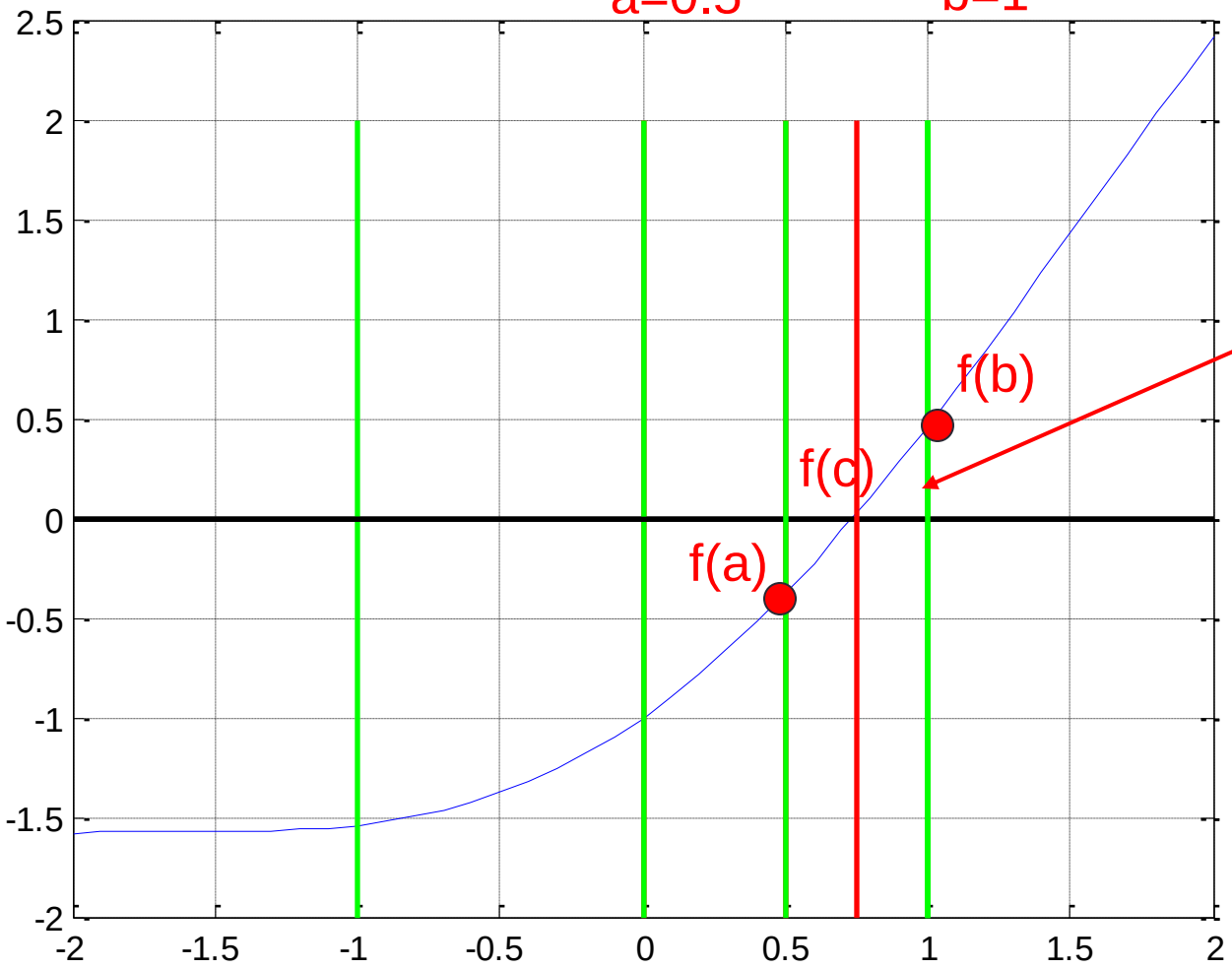


Primer
x-cos(x)=0 – Iteracija 2.



Primer
x-cos(x)=0 – Iteracija 3.

$$c = \frac{0.5+1}{2} = 0.75$$



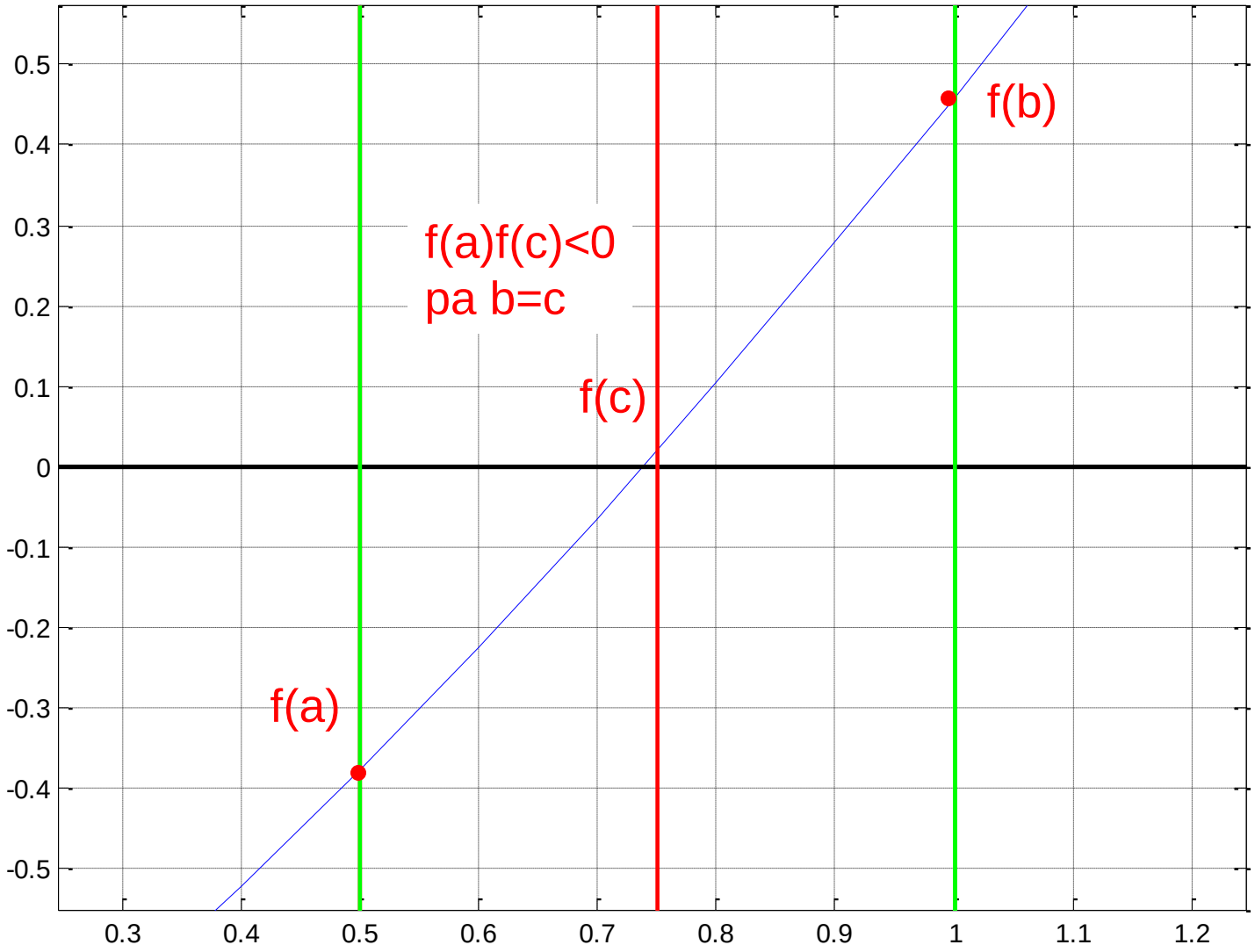
f(a)f(c)<0
pa b=c

videti

sledeći

slajd

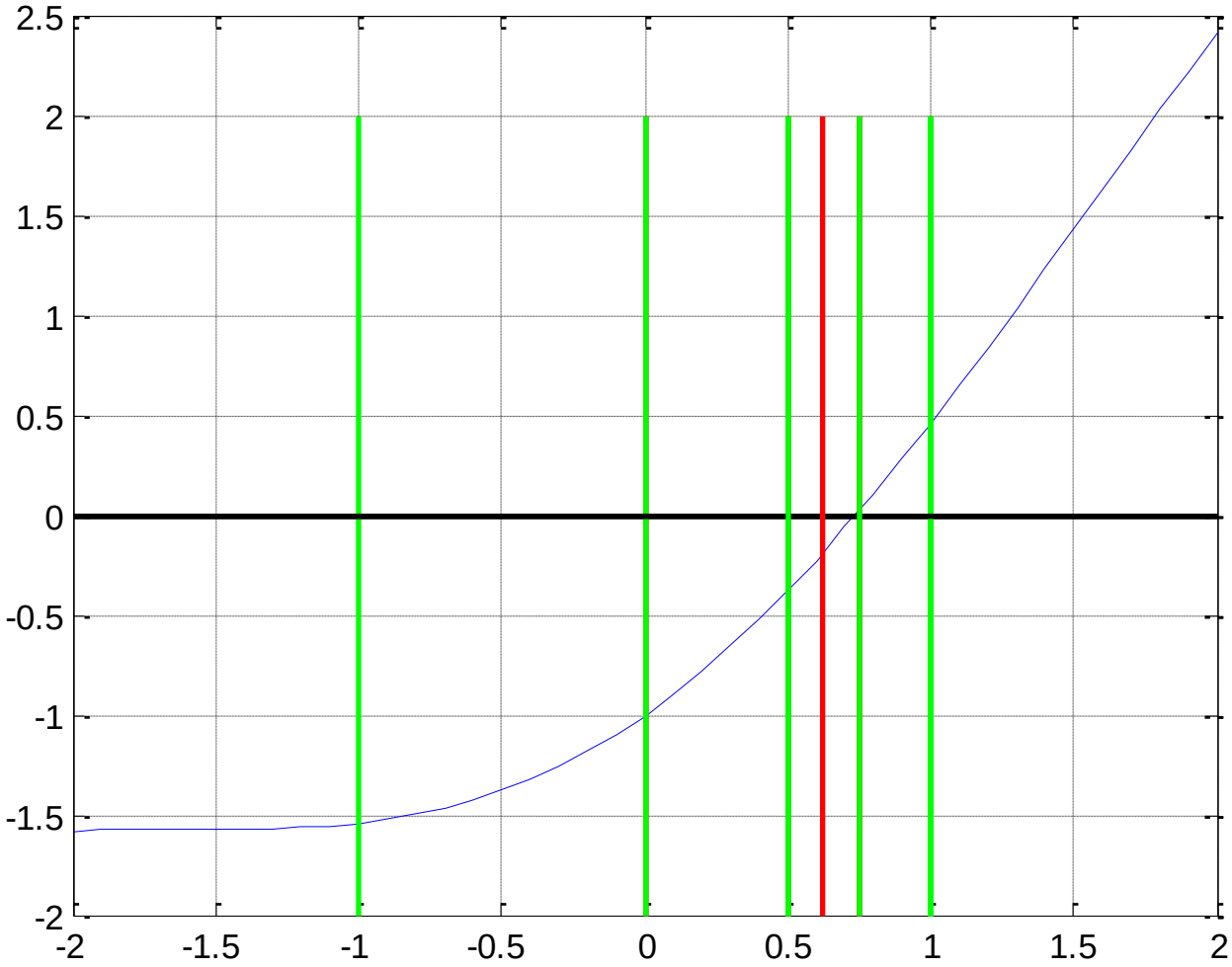
Primer
x-cos(x)=0 – Iteracija 3. Zoom



Primer
x-cos(x)=0 – Iteracija 4.

$$c = \frac{0.5 + 0.75}{2} = 0.625$$

a=0.5 b=0.75



Primer
x-cos(x)=0 – Tabela

Iteracija	*a*	*b*	*c*	$\lvert b-a \rvert$	*f(c)*
1	-1.000000	1.000000	0.000000	2.000000	-1.000000
2	0.000000	1.000000	0.500000	1.000000	-0.377583
3	0.500000	1.000000	0.750000	0.500000	0.018311
4	0.500000	0.750000	0.625000	0.250000	-0.185963
5	0.625000	0.750000	0.687500	0.125000	-0.085335
6	0.687500	0.750000	0.718750	0.062500	-0.033879
7	0.718750	0.750000	0.734375	0.031250	-0.007875
8	0.734375	0.750000	0.742188	0.015625	0.005196
9	0.734375	0.742188	0.738281	0.007813	-0.001345
10	0.738281	0.742188	0.740234	0.003906	0.001924

Posle 19 iteracija $\lvert b-a \rvert$ je 0.000008, a f(c) 0.000002

Metoda polovljenja konvergencija

- Ako je f(x) neprekidna i znamo da se rešenje nalazi u [a,b], metoda polovljenja garantovano konvergira.
- Konvergencija je spora.
- Pokazaćemo da je linearna.

Brzina konvergencije za iterativne metode

- Brzina konvergencije za iterativne metode:

Ako je tačno rešenje $\hat{x}$

Ako je greška u *k*-toj iteraciji $e^k = x^k - \hat{x}$

- Iterativna metoda konvergira brzinom *r* ako

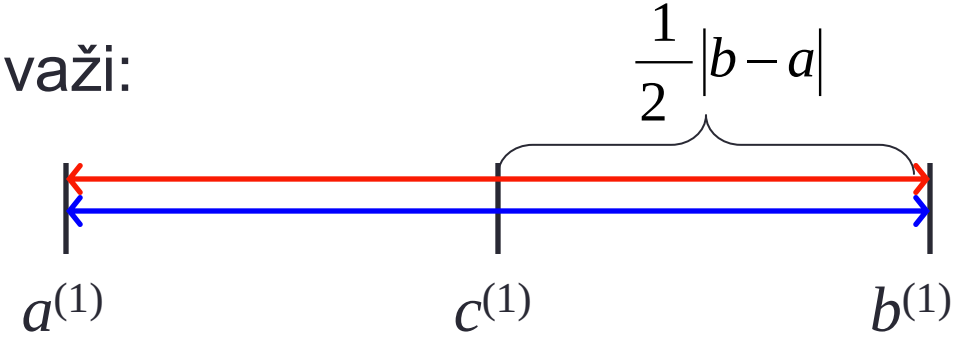
$$\lim_{k\to\infty} \frac{\left\| e^{k+1} \right\|}{\left\| e^k \right\|^r} = C$$

- r=1 i $C \in (0,1)$ – linearna konvergencija
- r=1 i $C = 0$ – superlinearna $k \to \infty \quad C \to 0$
- r=2 – kvadratna

Brzina konvergencije za metodu polovljenja

- Za grešku u k-toj iteraciji važi:

$$e^k = \left| x^k - \hat{x} \right| \leq \frac{1}{2^k} \left| b - a \right|$$



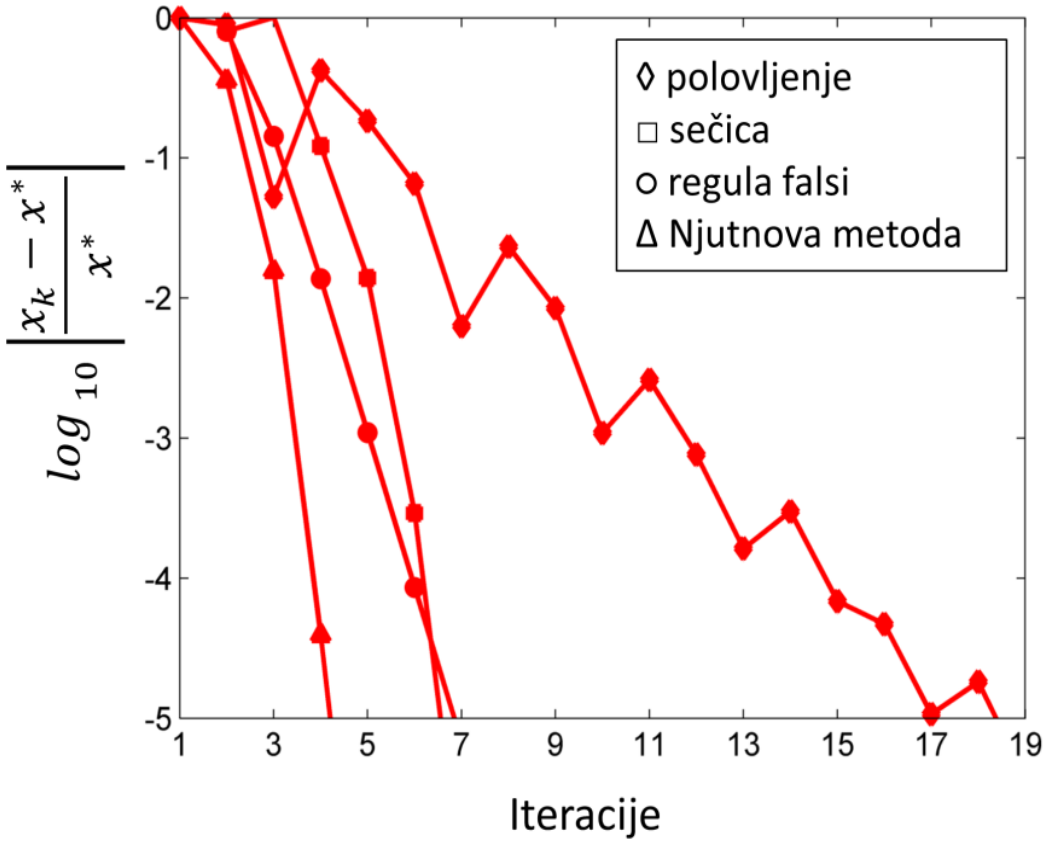
- pa za r=1

$$\lim_{k \to \infty} \frac{\left\| e^{k+1} \right\|}{\left\| e^k \right\|^r} = \frac{\frac{1}{2^{k+1}} \left| b - a \right|}{\frac{1}{2^k} \left| b - a \right|} = \frac{1}{2}$$

- što znači da je konvergencija linearna.

Konvergencija metoda, Grafički prikaz

Ilustracija brzine konvergencije različitih numeričkih metoda za problem:
x-cos(x)=0



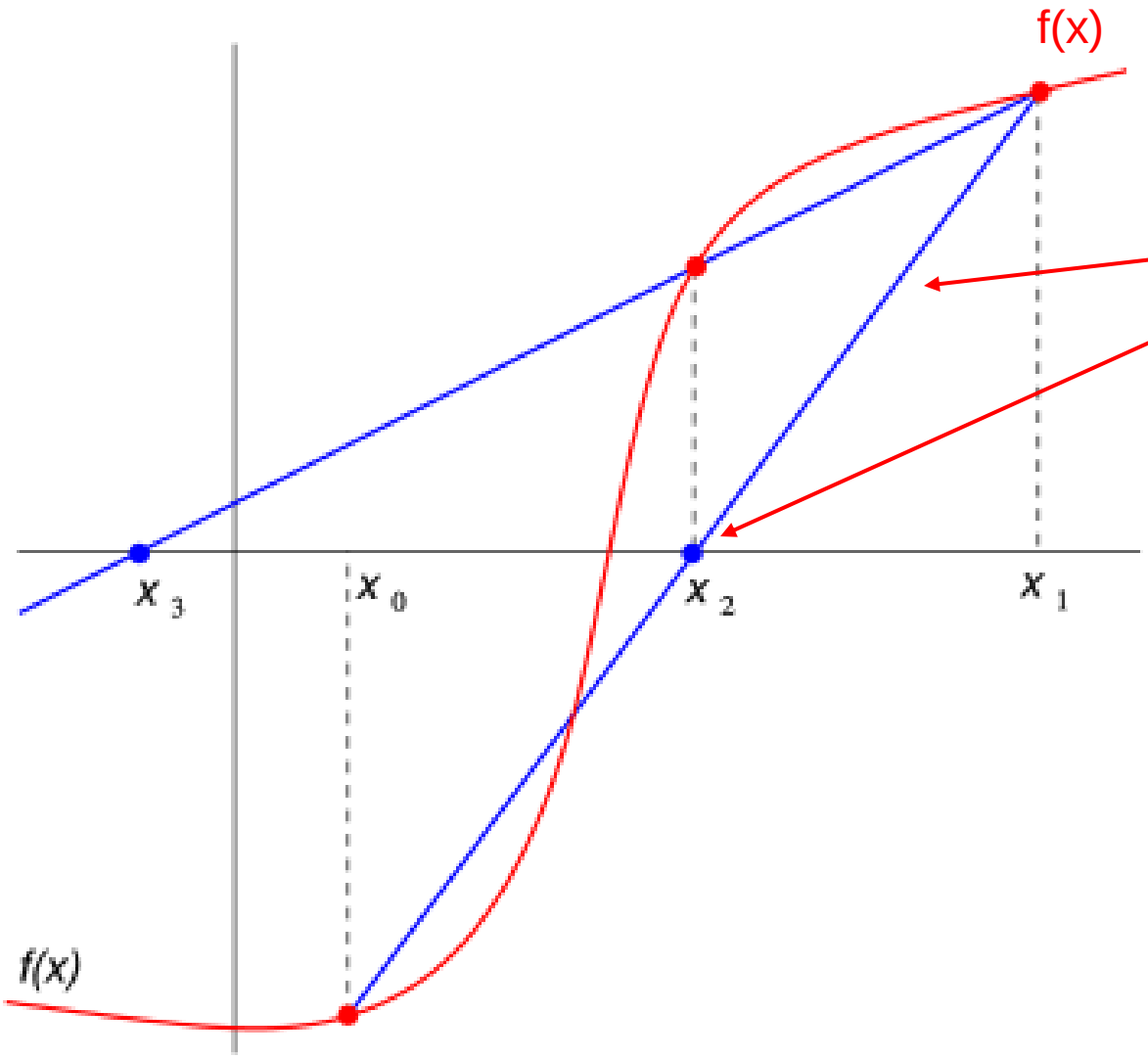
Metoda polovljenja
Rezime

- Prednosti:
  - uz odgovarajuće pretpostavke uvek konvergira
- Mane:
  - spora konvergencija
  - ignoriše vrednost funkcije, koristi samo interval [a,b]

Metoda sečice

- Videli smo da je mana metode polovljenja to što ne koristi vrednosti funkcije f(x) u svom algoritmu.
- Metoda sečice koristi funkciju f(x).
- Ideja je da prilikom traženja nule, f(x) zamenimo pravom na malom intervalu (linearna interpolacija).
- Umesto da tražimo nulu f(x) koja može biti komplikovanog oblika, tražimo nulu prave koja je jednostavna.

Metoda sečice



funkciju f(x), na intervalu $[x_0,x_1]$ zamenjujemo pravom (sečicom)

Nula sečice (x_2), nam je prva procena nule f(x).

Metoda sečice formula

Pretpostavke:

Dve početne tačke x_i i x_{i-1} takve da važi $f(x_i) \neq f(x_{i-1})$

Sledeća procena, tačka x_{i+1} dobija se pomoću formule:

$$x_{i+1} = x_i - f(x_i)\frac{(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

Metoda sečice izvođenje formule

- Treba nam prava kroz dve tačke x_i i x_{i-1}.
- x_{i+1} je tačka u kojoj ta prava seče x-osu tj. ima y=0.

$$y = kx + n \quad (x_{i-1}, f(x_{i-1})) \; i \; (x_i, f(x_i))$$

$$f(x_{i-1}) = kx_{i-1} + n$$

$$f(x_i) = kx_i + n \Rightarrow k = ? \; n = ?$$

$$(2) - (1) \Rightarrow f(x_i) - f(x_{i-1}) = k(x_i - x_{i-1}) \Rightarrow$$

$$k = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} ; n = f(x_i) - kx_i$$

$$zamenimo \; n \; u \; y = kx + n \Rightarrow y = kx + f(x_i) - kx_i \Rightarrow$$

$$y - f(x_i) = k(x - x_i) \Rightarrow y - f(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}(x - x_i)$$

Da bi dobili x_{i+1} iz x_i i x_{i-1} koristimo jednačinu prave kroz tačke x_i and x_{i-1}.

Onda u nju zamenimo $(x_{i+1},0)$ i rešimo za x_{i+1}.

$$y - f(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}(x - x_i)$$

jednačina prave

$$0 - f(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}(x_{i+1} - x_i)$$

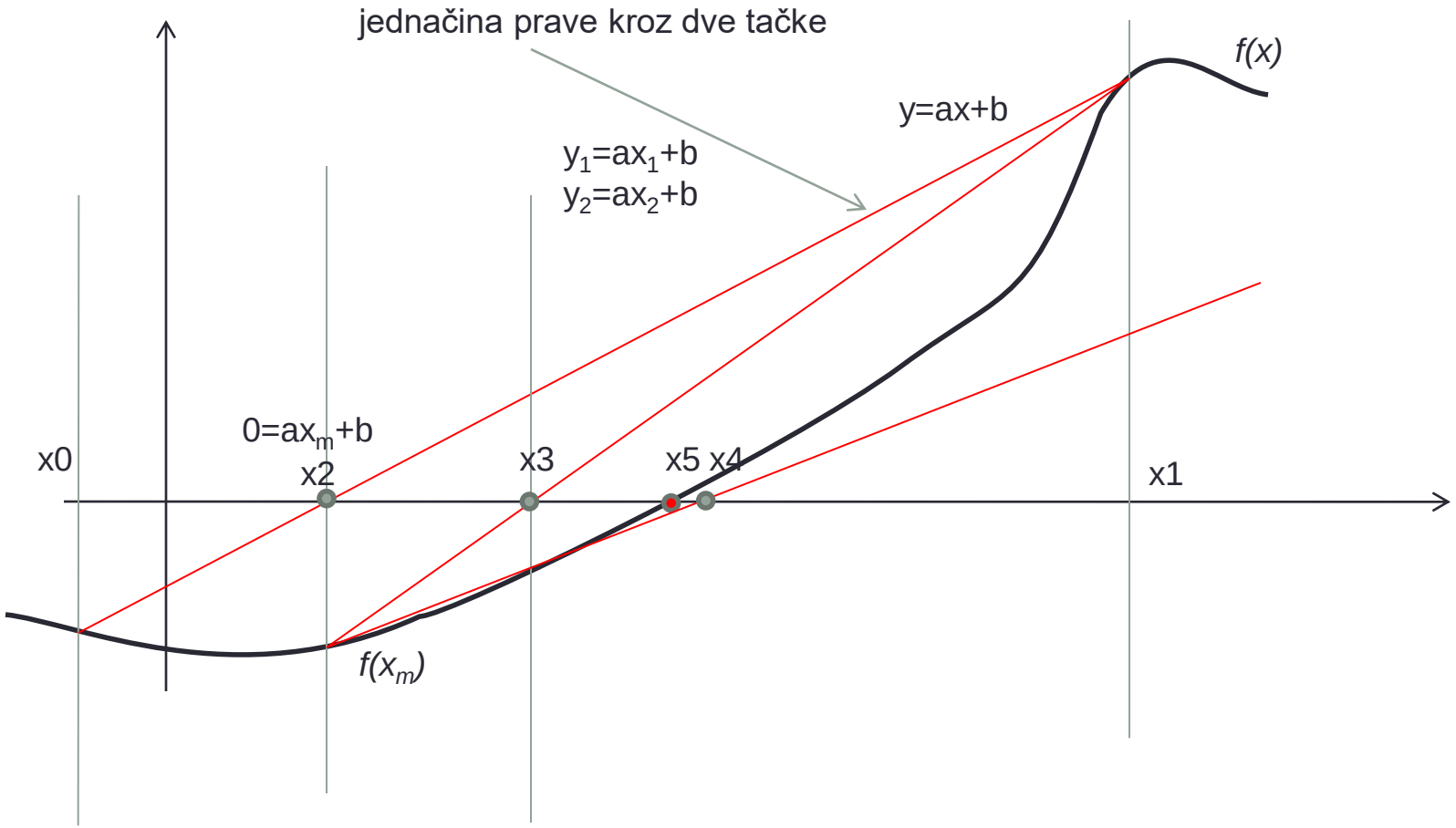
zamenimo $(x_{i+1},0)$

$$\frac{-f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} = x_{i+1} - x_i$$

rešimo za x_{i+1}

$$x_{i+1} = x_i - f(x_i)\frac{(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

Metoda sečice



Metoda sečice
Napomena

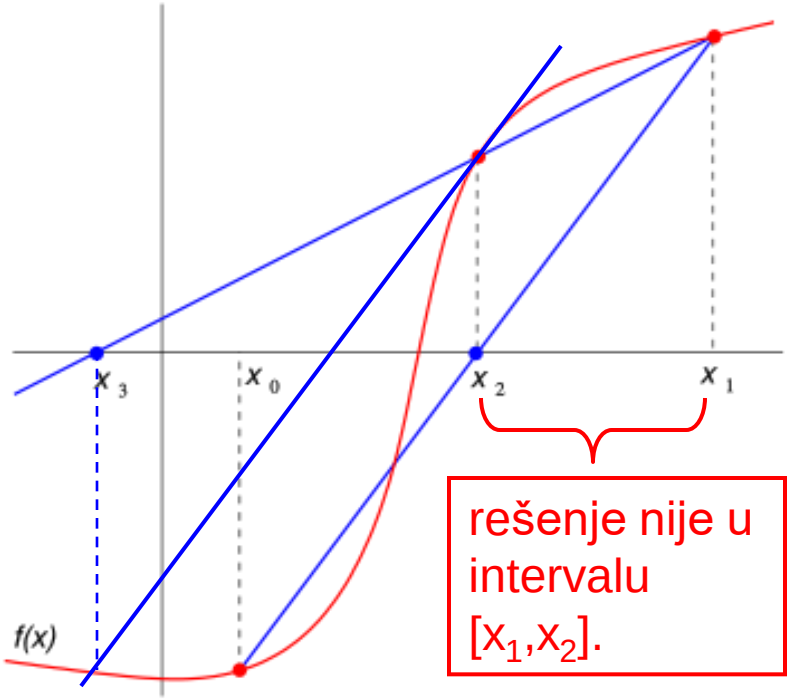
- Metoda sečice je otvorena metoda.
- Ne zahteva da za prve dve tačke važi $f(x_0)f(x_1)<0$ tj.
- Ne zahteva da se rešenje nalazi u intervalu $[x_0,x_1]$.

$$x_2 = x_1 - f(x_1)\frac{(x_1 - x_0)}{f(x_1) - f(x_0)}$$

$$x_3 = x_2 - f(x_2)\frac{(x_2 - x_1)}{f(x_2) - f(x_1)}$$

$$x_4 = x_3 - f(x_2)\frac{(x_3 - x_2)}{f(x_3) - f(x_2)}$$

....



Matlab kod

```
function x=secica(a,b,maxIter,tacnost,funkcija)
for i=1:maxIter
  fa=feval(funkcija,a);
  fb=feval(funkcija,b);
  k=(fb-fa)/(b-a);
  n=fb-k*b;
  c=-n/k;
  fc=feval(funkcija,c);
  if(abs(fc)<tacnost)
        break;
   end
   a=b;
   b=c;
end
x=c;
```

sečica je otvoren metod pa se ne proverava f(a)f(b)<0

y=kx+n , za y=0, x=? ->

0=kx+n -> x=-n/k

sečica je otvoren metod pa se abs(b-a) ne smanjuje dovoljno brzo sa približavanjem rešenju.

Zato koristimo smanjenje vrednosti funkcije fc kao uslov zaustavljanja.

Matlab kod

- Specificiramo jednačinu koju rešavamo tj. funkciju čiju nulu tražimo.
- Dva od mogućih načina:
  - kreiramo posebnu Matlab funkciju

```
function y = jednacina(x)
y=x-cos(x);
```

  - `secica(-1,1,100,10^-5,'jednacina')`
  - prosleđujemo funkciju direktno u pozivu metode polovljenja
  - `secica(-1,1,100,10^-5,@(x)x-cos(x))`

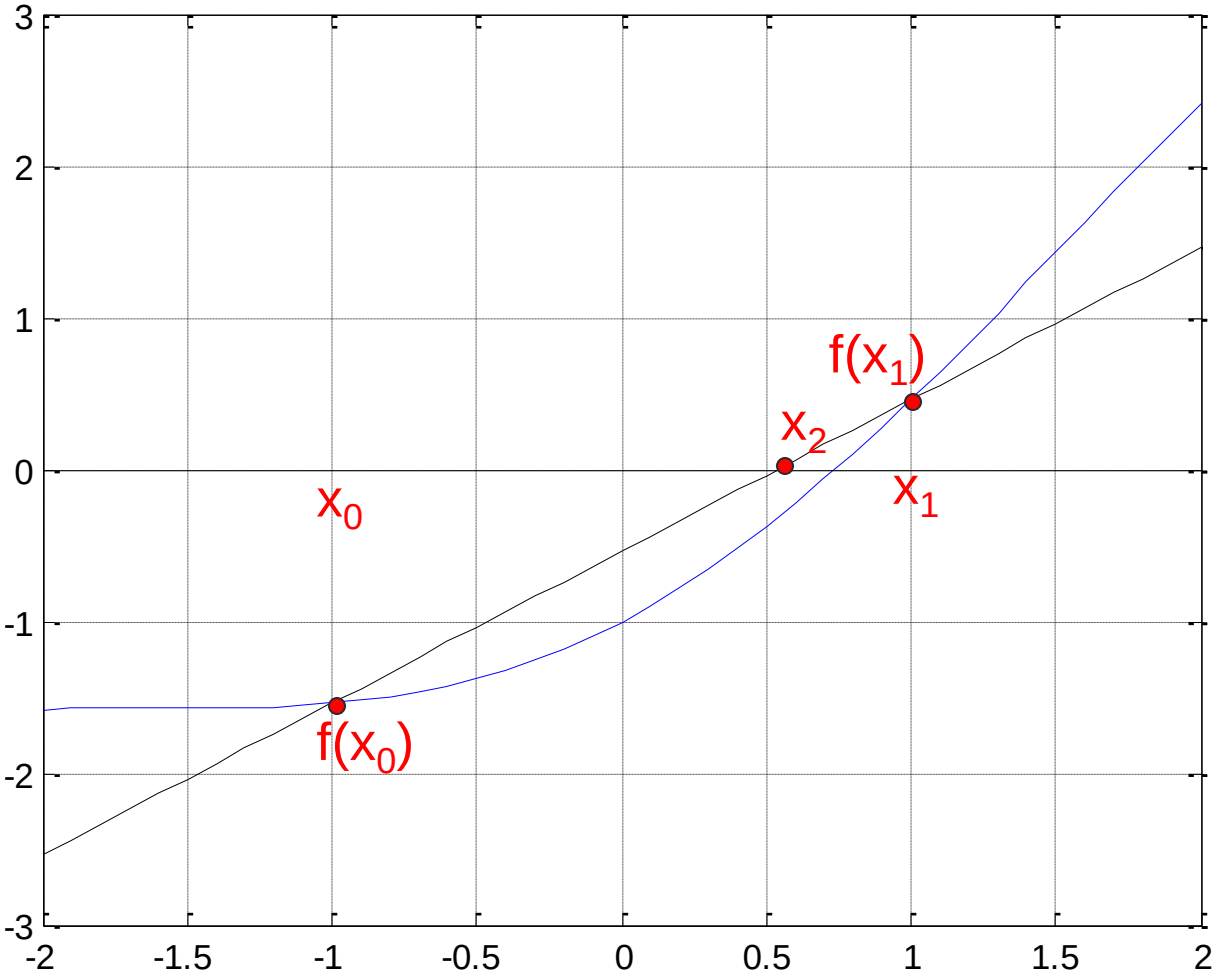
Primer

x-cos(x)=0 – Iteracija 1.

x0=-1

$$x_2 = x_1 - f(x_1)\frac{(x_1 - x_0)}{f(x_1) - f(x_0)} = 0.540302$$

x1=1



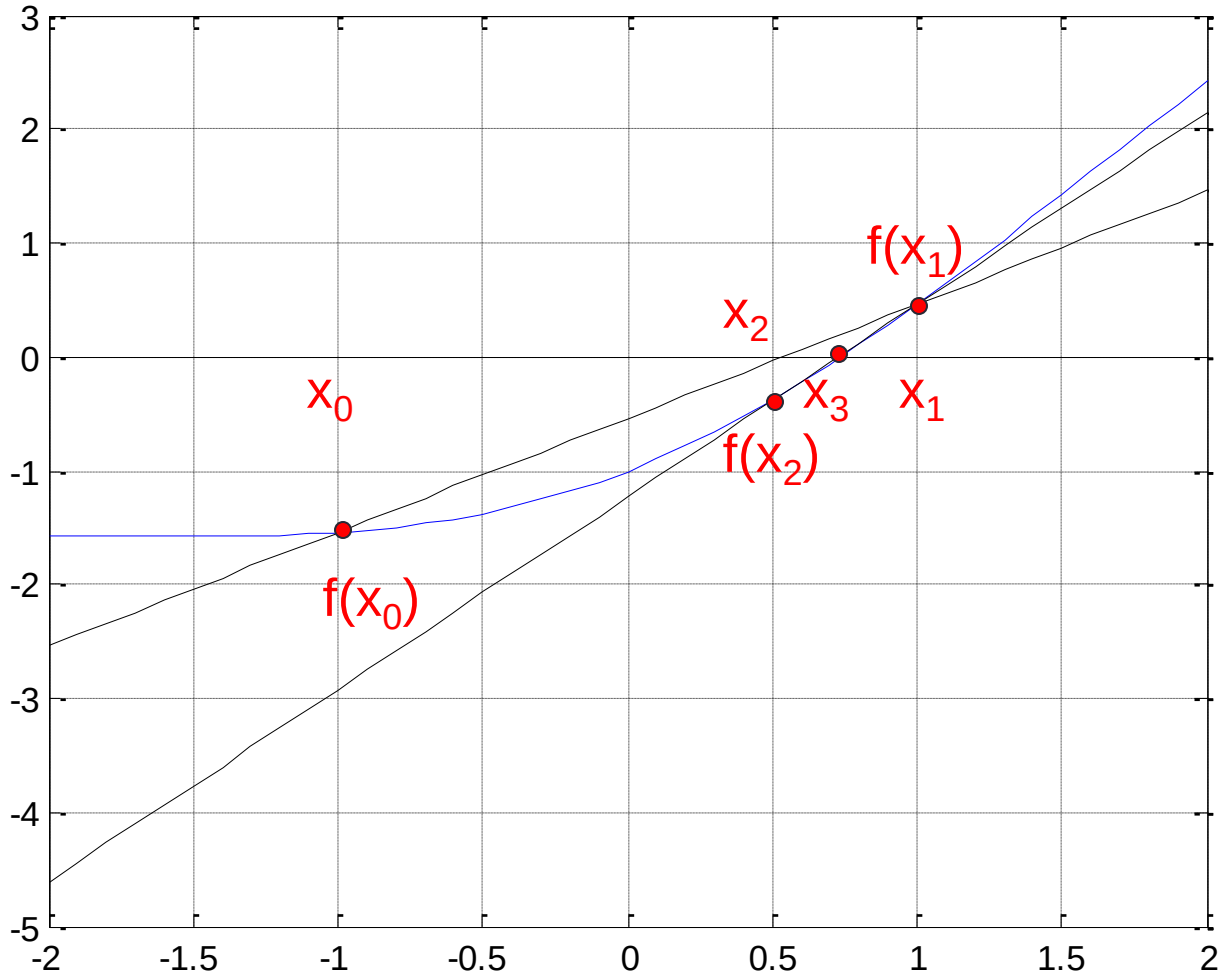
Primer

x-cos(x)=0 – Iteracija 2.

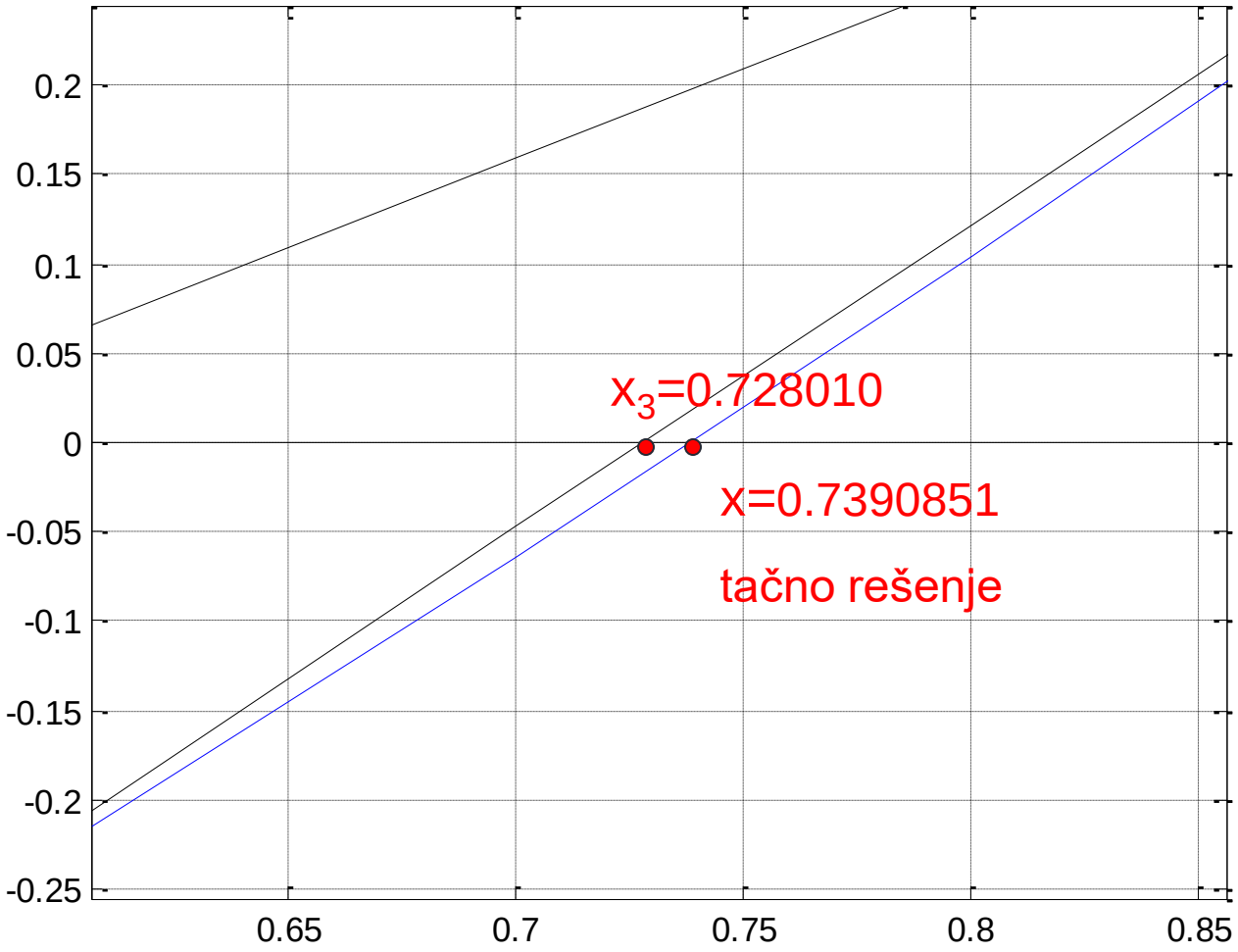
x0=-1

$$x_3 = x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)} = 0.728010$$

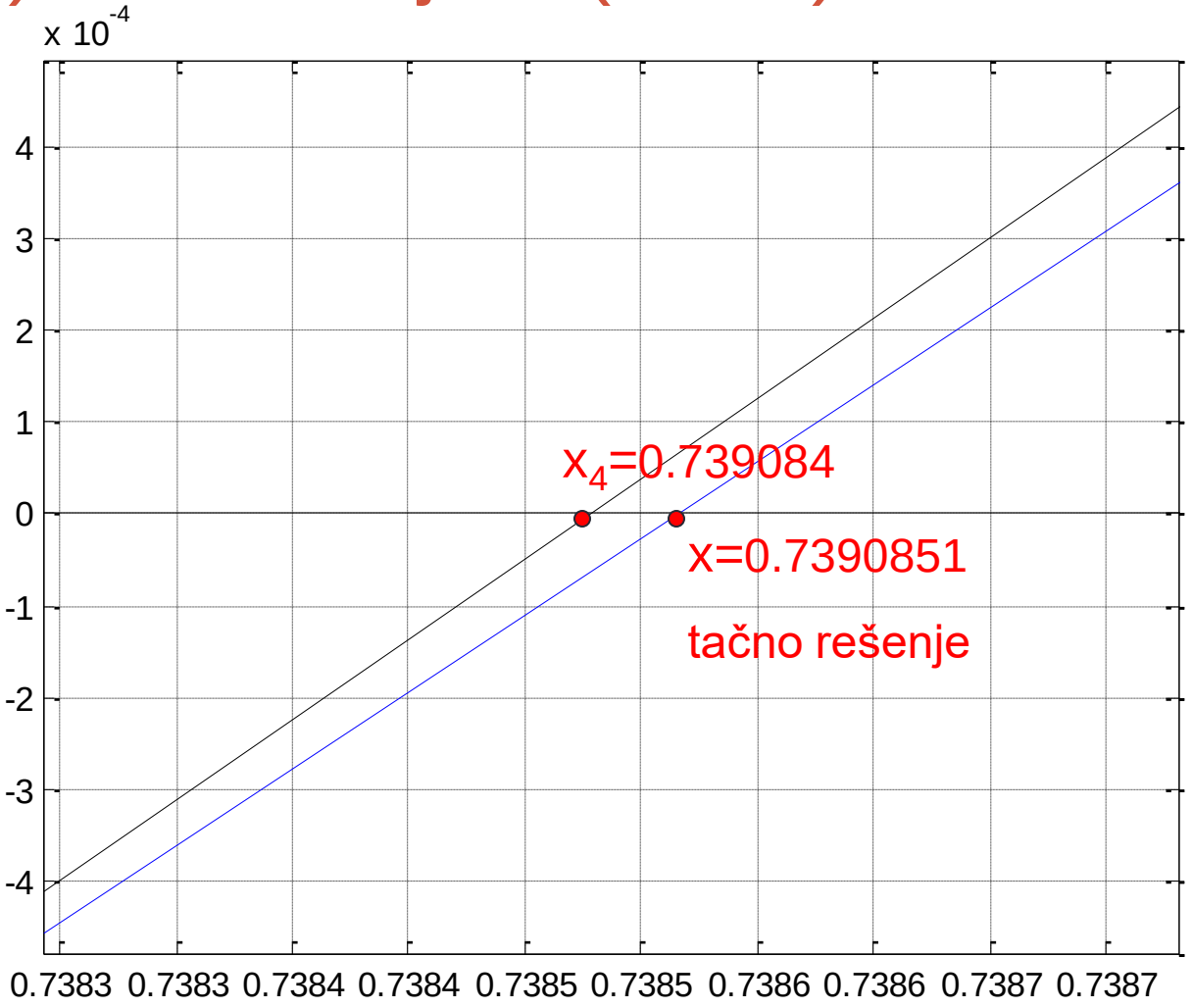
x1=1



Primer
x-cos(x)=0 – Iteracija 2. (Zoom)



Primer
x-cos(x)=0 – Iteracija 3. (Zoom)



Napomena:

x_4 deluje da je udaljeniji od x u odnosu na x_3 (prethodni slajd), samo zato što je na ovom slajdu veći zoom.

Primer
x-cos(x)=0 interval x_0=-1, x_1=1
Tabelarni prikaz

Iteracija	*a*	*b*	*c*	$\lvert b-a \rvert$	$\lvert f(c) \rvert$
1	-1.000000	1.000000	0.540302	2.00000000000000000	0.31725090997825367
2	1.000000	0.540302	0.728010	0.45969769413186023	0.01848939457760324
3	0.540302	0.728010	0.739627	0.18770805559947734	0.00090700440040725
4	0.728010	0.739627	0.739084	0.01161665116311650	0.00000222997338051

Posle 4 iteracije $\lvert f(c) \rvert$ je 0.000002.

Kod polovljenja to je bio slučaj tek posle 19 iteracija.

Primer
x-cos(x)=0 interval x_0=1, x_1=2
– Tabelarni prikaz

Rekli smo da je sečica otvoren metod, tj. da rešenje ne mora da bude u intervalu između početnih tačaka.

Na primeru datom u tabeli nalaze se rezultati metode sečice za x_0=1 i x_1=2.

Podsetimo se tačno rešenje je x=0.7390851 i vidimo da $x \notin [x_0, x_1]$

Iteracija	*a*	*b*	*c*	$\lvert b-a \rvert$	$\lvert f(c) \rvert$
1	1.000000	2.000000	0.765035	1.00000000000000000	0.04367634422860511
2	2.000000	0.765035	0.742299	1.23496531760818100	0.00538326126319744
3	0.765035	0.742299	0.739103	0.02273527552687527	0.00003035432883436
4	0.742299	0.739103	0.739085	0.00319613670600805	0.00000002150675038

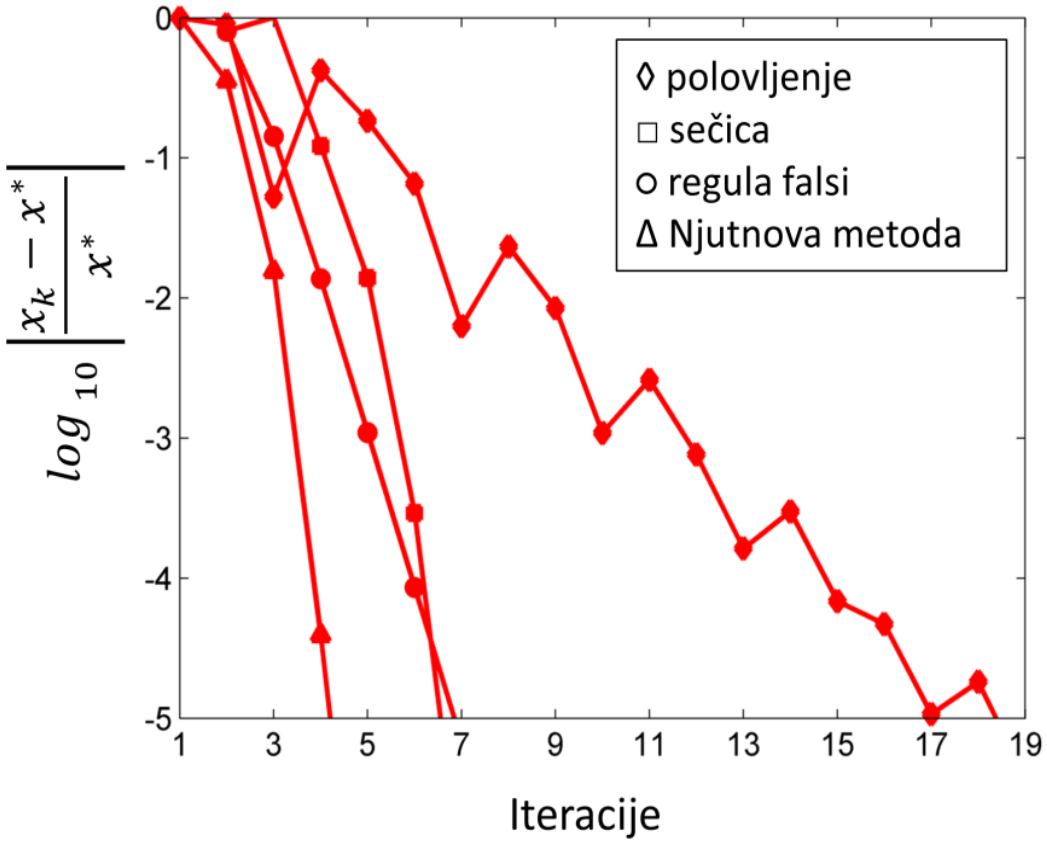
Posle 4 iteracije $\lvert f(c) \rvert$ je 0.00000002.

Metoda sečice
Rezime

- Prednosti:
  - brzo konvergira (superlinearno).
  - ne zahteva da se rešenje nalazi u početnom intervalu.
- Mane:
  - konvergencija nije uvek garantovana.
    - uslov je da su x_0 i x_1 “dovoljno blizu” tačnog rešenja.
  - broj iteracija potreban za toleranciju ne može se znati unapred

Konvergencija metoda, Grafički prikaz

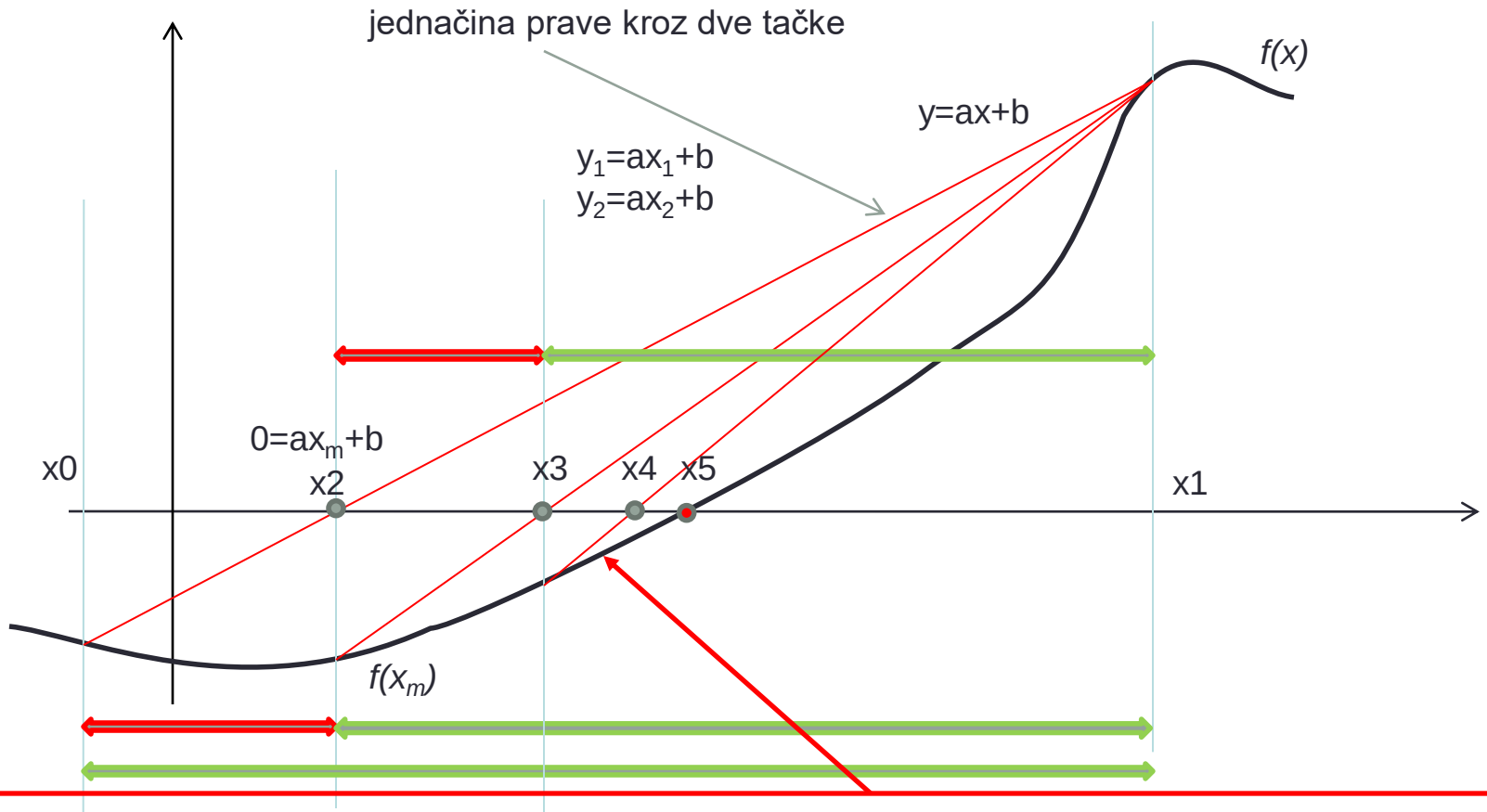
Ilustracija brzine konvergencije različitih numeričkih metoda za problem:
x-cos(x)=0



Metoda regula falsi (*false position*)

- Videli smo da metoda sečice nema garantovanu konvergenciju jer ne zahteva da se rešenje nalazi u zatvorenom intervalu.
- Metoda regula falsi uvodi taj zahtev u metodu sečice.
  - Zahteva i da na početku važi $f(a)f(b)<0$.
- Time dobijamo metodu koja malo sporije ali garantovano konvergira.

Metoda regula falsi



kod metode sečice prava za x4 išla je od (x2,f(x2)) do (x3,f(x3)) i važi f(x2)f(x3)>0

Sada međutim prava u istom koraku ide od (x1,f(x1)) (x3,f(x3)) jer u tom intervalu važi f(x1)f(x3)<0

Matlab kod

```
function x=regula_falsi(a,b,maxIter,tacnost,funkcija)
if(feval(funkcija,a)*feval(funkcija,b)<0)
    for i=1:maxIter
     fa=feval(funkcija,a);
     fb=feval(funkcija,b);
     k=(fb-fa)/(b-a);
     n=fb-k*b;
     c=-n/k;
     fc=feval(funkcija,c);
     if(abs(fc)<tacnost)
         break;
     end
     if(fc*fa<0)
         b=c;
     else
         a=c;
     end
    end
    x=c;
end
```

Ovo je zatvoren metod pa zahteva f(a)f(b) <0

Kod metoda sečice bilo je samo:
a=b;
b=c;

Primer
x-cos(x)=0 x_0=-1, x_1=1
Tabelarni prikaz

Iteracija	*a*	*b*	*c*	$\lvert b-a \rvert$	$\lvert f(c) \rvert$
1	-1.000000	1.000000	0.540302	2.00000000000000000	0.31725090997825367
2	0.540302	1.000000	0.728010	0.45969769413186023	0.01848939457760324
3	0.728010	1.000000	0.738527	0.27198963853238289	0.00093397288128583
4	0.738527	1.000000	0.739057	0.26147299375760025	0.00004680484352704
5	0.739057	1.000000	0.739084	0.26094283332173240	0.00000234462403403

Posle 5 iteracija $\lvert f(c) \rvert$ je 0.000002.

Kod metode sečice to je bio slučaj posle 4 iteracije.

Poređenje sa metodom sečice

Metoda regula falsi

Iteracija	*a*	*b*	*c*	f(a)*f(b)	$\lvert f(c) \rvert$
1	-1.000000	1.000000	0.540302	-0.70807341827357118	0.31725090997825367
2	0.540302	1.000000	0.728010	-0.14583951177823759	0.01848939457760324
3	0.728010	1.000000	0.738527	-0.00849953205321833	0.00093397288128583
4	0.738527	1.000000	0.739057	-0.00042934517990878	0.00004680484352704
5	0.739057	1.000000	0.739084	-0.00002151607864358	0.00000234462403403

Metoda sečice

Iteracija	*a*	*b*	*c*	f(a)*f(b)	$\lvert f(c) \rvert$
1	-1.000000	1.000000	0.540302	-0.70807341827357118	0.31725090997825367
2	1.000000	0.540302	0.728010	-0.14583951177823759	0.01848939457760324
3	0.540302	0.728010	0.739627	0.00586577725469162	0.00090700440040725
4	0.728010	0.739627	0.739084	-0.00001676996224275	0.00000222997338051

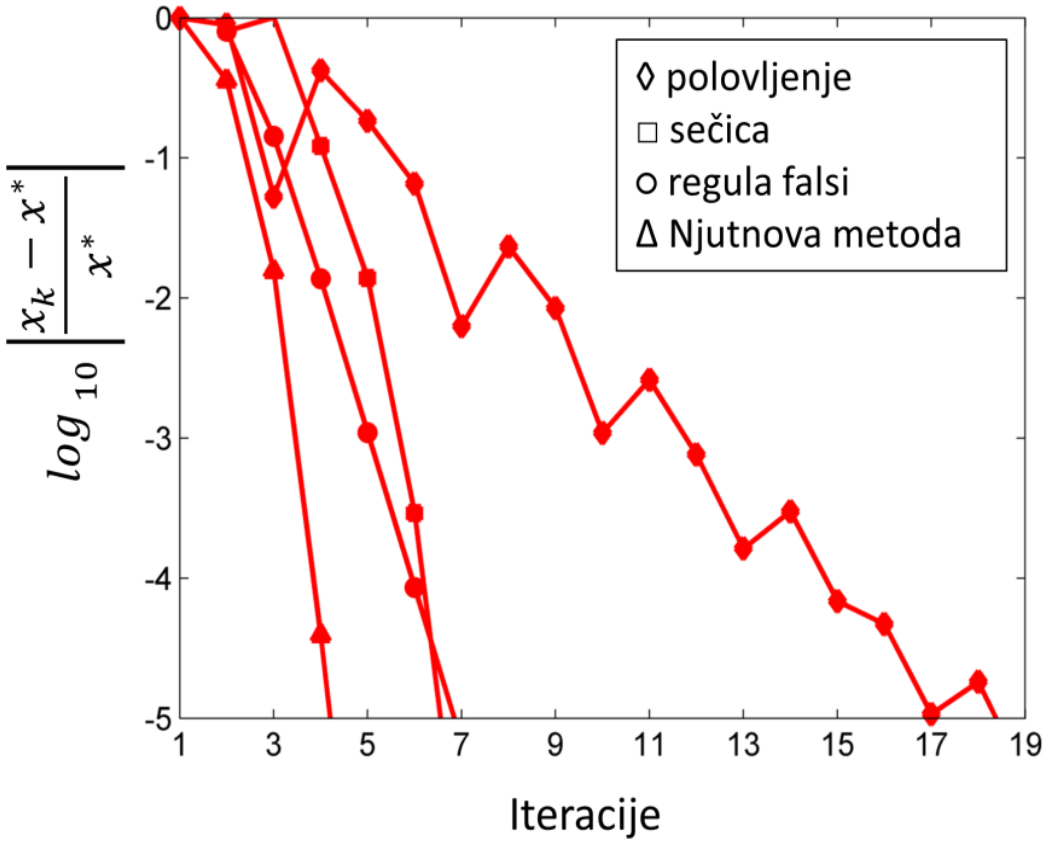
U iteraciji 3. metoda sečice ne nastavlja u intervalu kod koga je f(a)f(b)<0, što je u ovom slučaju ubrzalo konvergenciju.

Metoda regula falsi
Rezime

- Prednosti:
  - garantovana konvergencija (brža od polovljenja)
- Mane:
  - zahteva da se rešenje nalazi u početnom intervalu tj. $f(a)f(b)<0$
  - sporija konvergencija od metode sečice

Konvergencija metoda, Grafički prikaz

Ilustracija brzine konvergencije različitih numeričkih metoda za problem:
x-cos(x)=0



Njutnova metoda (metoda tangente)

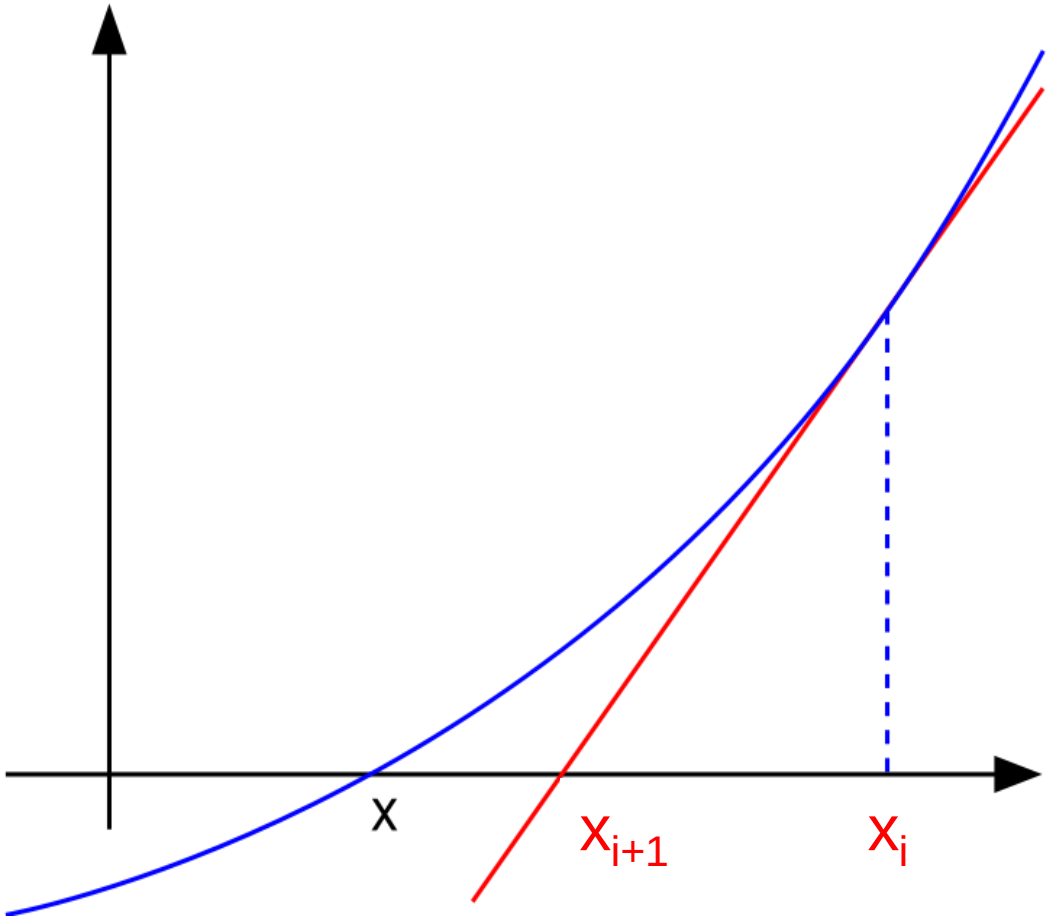
- Videli smo da metoda sečice funkciju f(x) aproksimira pravom.
- Metoda Njutna koristi istu ideju ali je unapređuje.
- Funkcija f(x) aproksimira se tangentom.
- Poenta je u tome da je tangenta bolja aproksimacija tj. bolje “prati” funkciju f, u odnosu na sečicu.

Njutnova metoda (metoda tangente)

- Pretpostavke
  - f(x) je neprekidna i ima prvi izvod
  - x_0 je početna tačka takva da je $f'(x_0) \neq 0$
- Sledeća tačka računa se po formuli:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Njutnova metoda (metoda tangente)



jednačina tangente u tački x_i:

$$y = f(x_i) + f'(x_i)(x - x_i)$$

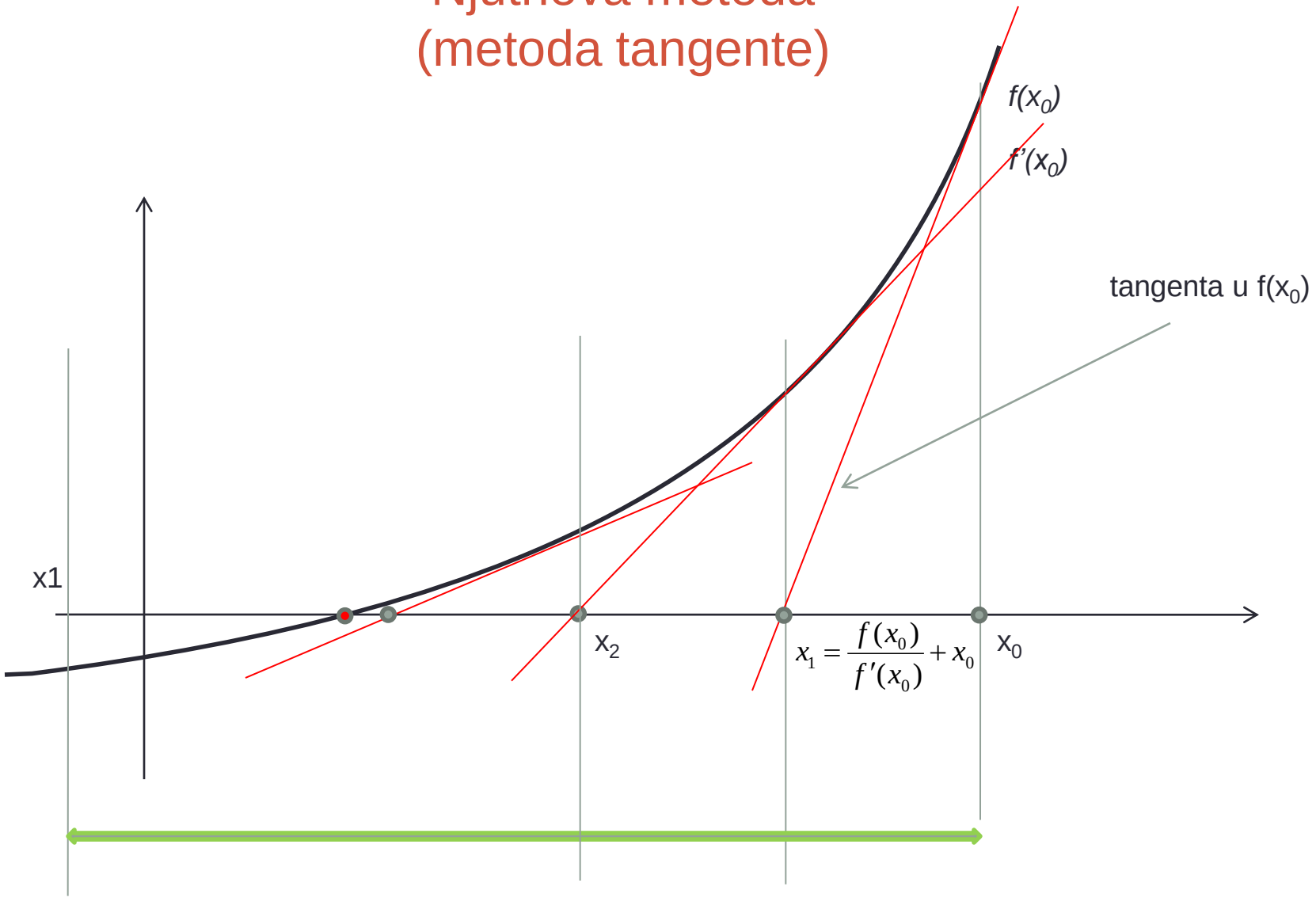
tangenta seče x-osu u tački x_{i+1} za y=0:

$$0 = f(x_i) + f'(x_i)(x - x_i)$$

$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$

$$\Rightarrow x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Njutnova metoda (metoda tangente)



Matlab kod

```
function x=tangenta(maxIter,tacnost,funkcija,izvod,x0)
xn=x0;
xnplus1=x0;
    for i=1:maxIter
        fn = feval(funkcija,xn);
        xnplus1 = xn-(fn/feval(izvod,xn));
        fnplus1 = feval(funkcija,xnplus1);
        if(abs(fnplus1)<tacnost)
            break;
        end
        xn=xnplus1;
    end
x=xnplus1;
```

Matlab kod

- Dva od mogućih načina da zadamo izvod:
  - kreiramo još jednu Matlab funkciju

```
function y = jednacina(x)
y=x-cos(x);
```

```
function y=jednacina_izvod(x)
y=1+sin(x);
```

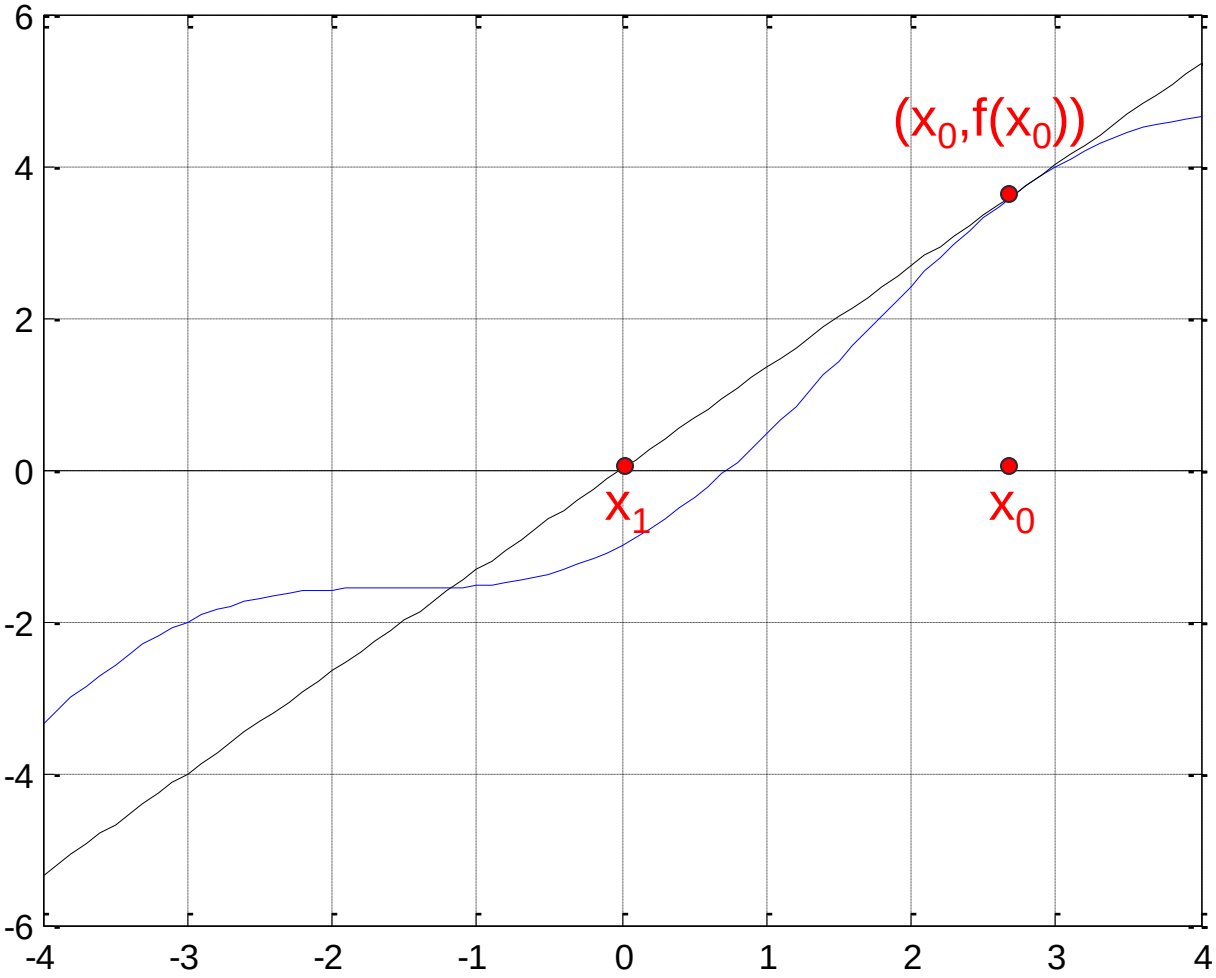
  - `tangenta(100,10^-5,'jednacina','izvod',2.8)`
  - prosleđujemo funkciju direktno u pozivu metode polovljenja
  - `tangenta(100,10^-5,@(x)x-cos(x),@(x)1+sin(x),2.8)`
- Simboličko određivanje izvoda u Matlabu:

```
>> syms x
>> diff('x-cos(x)')
ans =
1+sin(x)
```

Primer
x-cos(x)=0 x_0=2.8

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 2.8 - \frac{2.8 - \cos(2.8)}{1 + \sin(2.8)} = -0.003188$$

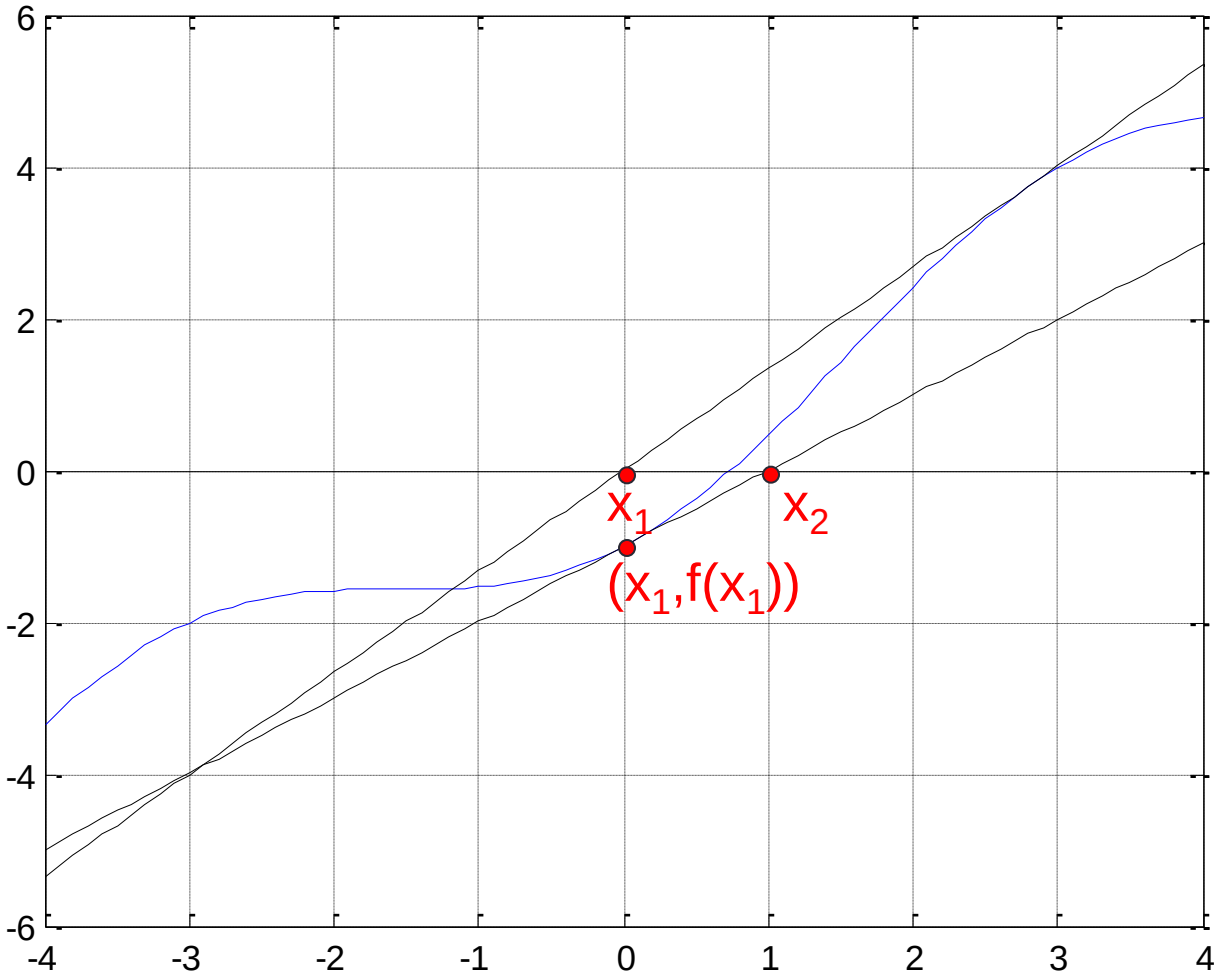
Iteracija 1.



Napomena: x_0=2.8 (a ne x_0=1 ili x_0=-1 kao ranije) odabran je da bi se bolje ilustrovala metoda.

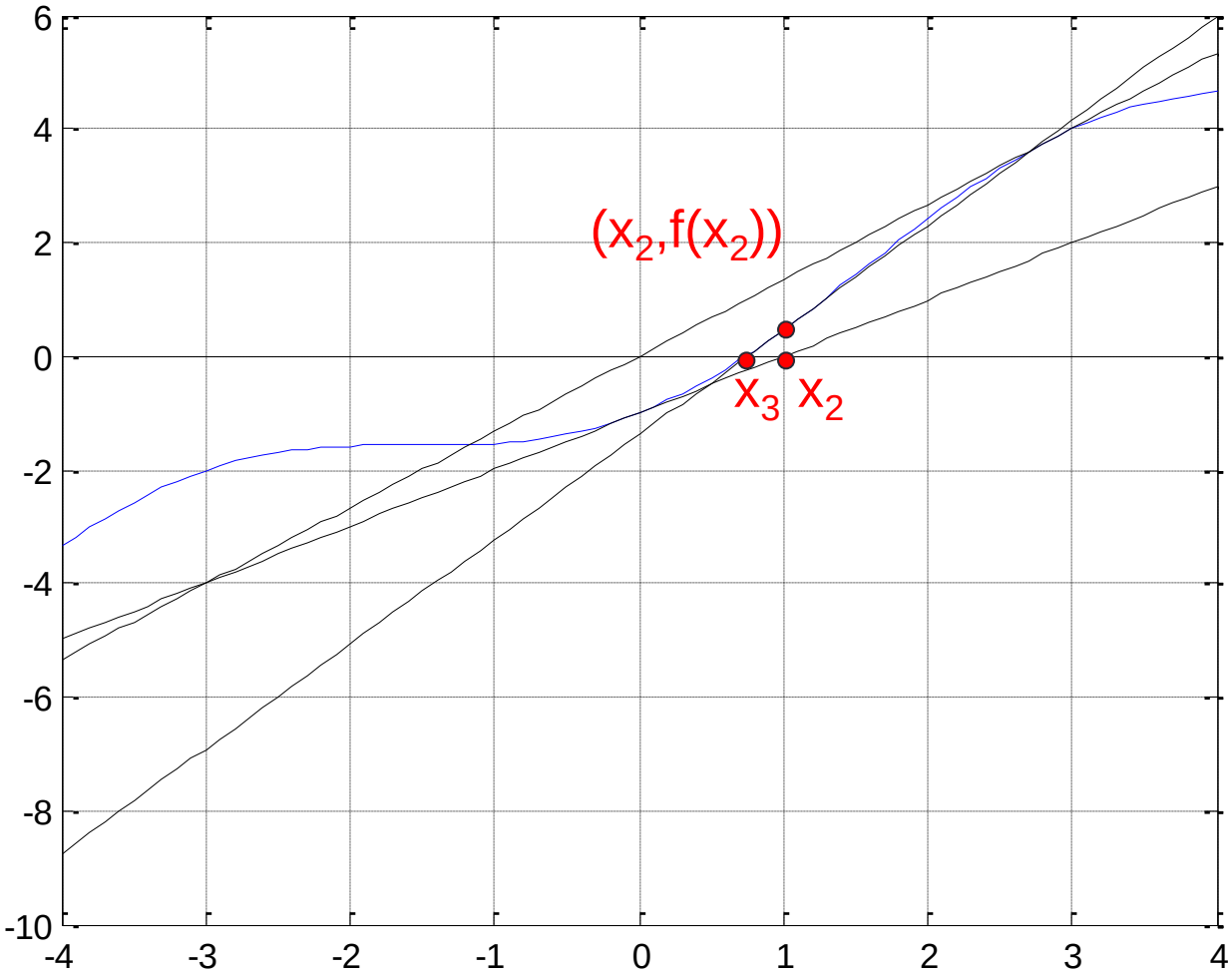
$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = -0.003188 - \frac{-0.003188 - \cos(-0.003188)}{1 + \sin(-0.003188)} = 1.003203$$

Iteracija 2.



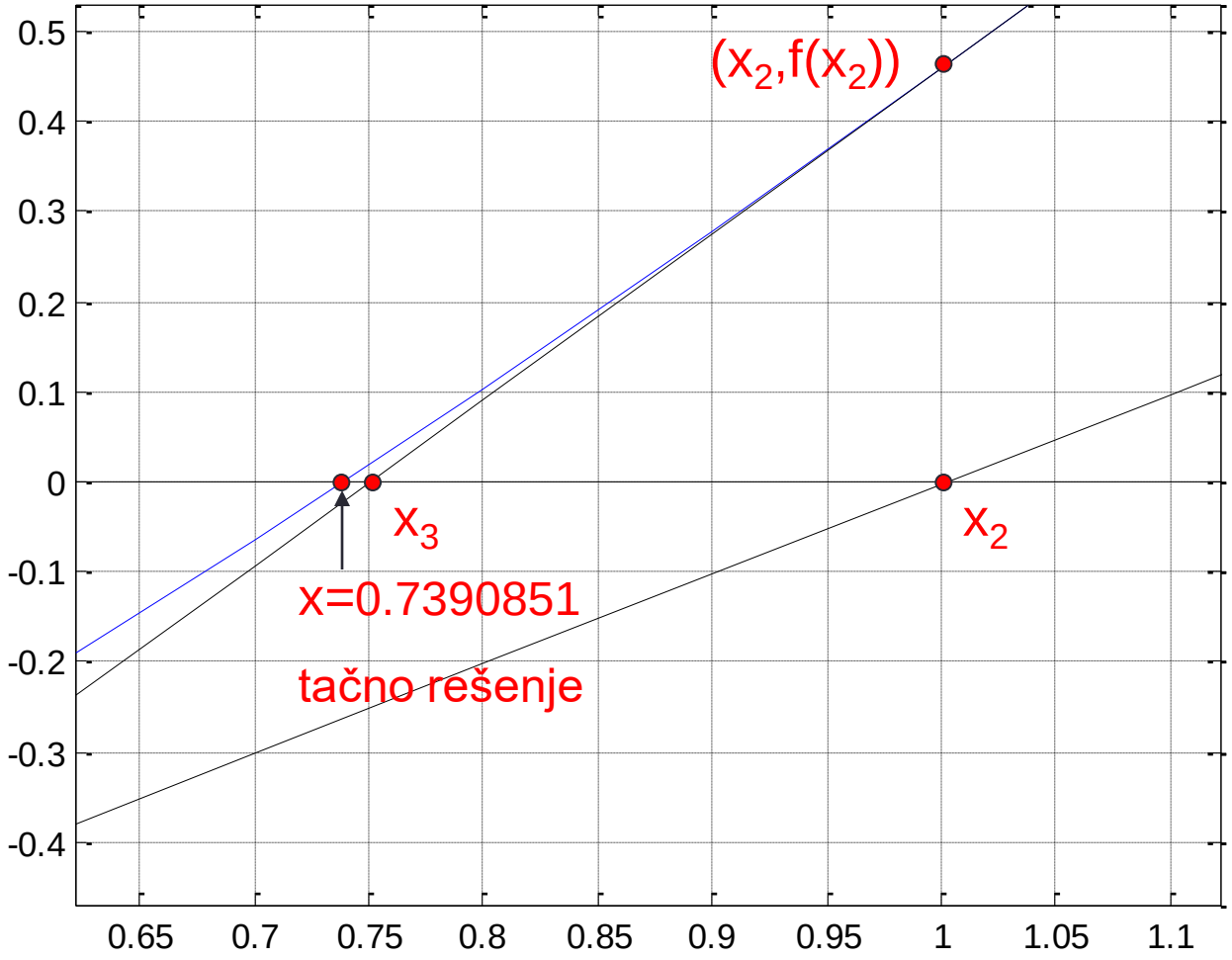
$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = 1.003203 - \frac{1.003203 - \cos(1.003203)}{1 + \sin(1.003203)} = 0.750599$$

Iteracija 3.



$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = 1.003203 - \frac{1.003203 - \cos(1.003203)}{1 + \sin(1.003203)} = 0.750599$$

Iteracija 3. (Zoom)



Primer
x-cos(x)=0 x_0=2.8
Tabelarni prikaz

Iteracija	x_n	x_{n+1}	$\lvert f(x_{n+1}) \rvert$
1	2.800000	-0.003188	1.00318260323936760
2	-0.003188	1.003203	0.46559863583045469
3	1.003203	0.750599	0.01931883983478933
4	0.750599	0.739114	0.00004840246208071
5	0.739114	0.739085	0.00000000030908021

Posle 5 iteracija $\lvert f(c) \rvert$ je 0.0000000003

Metoda Njutna u ovom slučaju konvergira sporije od metode sečice samo zato što je početno rešenje 2.8, dok je kod sečice bilo x_1=1 (što je mnogo bliže pravoj nuli).

Primer metode Njutna za x_0=1 dat je na sledećem slajdu.

Primer
x-cos(x)=0 x_0=1
Tabelarni prikaz

Iteracija	x_n	x_{n+1}	$\lvert f(x_{n+1}) \rvert$
1	1.000000	0.750364	0.01892307382211744
2	0.750364	0.739113	0.00004645589899077
3	0.739113	0.739085	0.00000000028472058

Posle 3 iteracije $\lvert f(c) \rvert$ je 0.0000000002

Kod metode sečice to je bio slučaj posle 4 iteracije.

Njutnova metoda Konvergencija

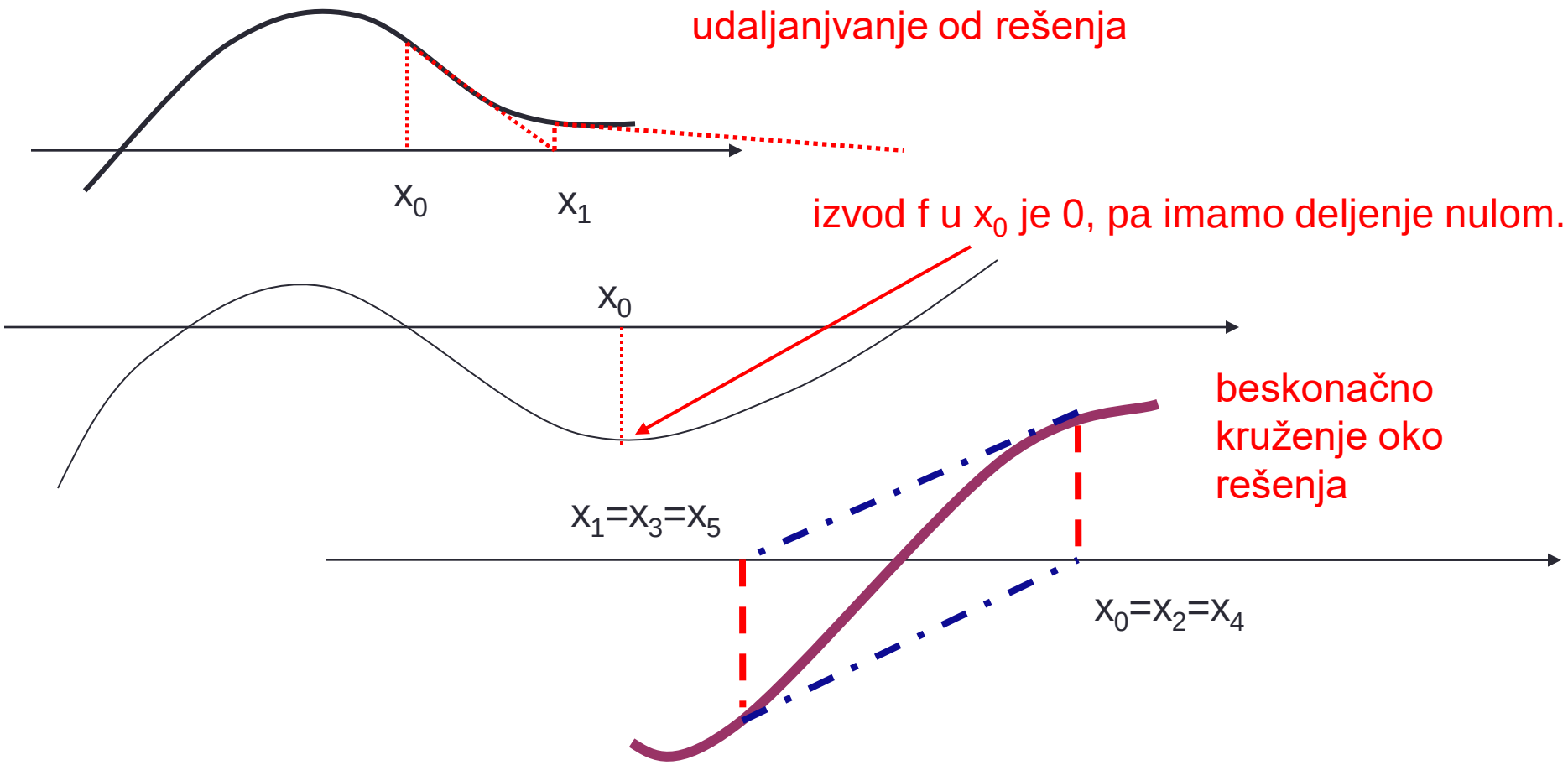
- Ako važi da su $f(x),\ f'(x)\ i\ f''(x)$ neprekidni blizu nule funkcije r i ako je $f'(r) \neq 0$ onda je:
  - Konvergencija Njutnovog metoda kvadratna.
  - Kvadratna konvergencija znači da se broj tačnih cifara rešenja duplira u svakoj iteraciji.

Iteracija	x_n	x_{n+1}	$\lvert f(x_{n+1}) \rvert$
1	1.00000000000000000	0.75036386784024389	0.01892307382211744
2	0.75036386784024389	0.73911289091136168	0.00004645589899077
3	0.73911289091136168	0.73908513338528403	0.00000000028472058

x=0.7390851
tačno rešenje

Njutnova metoda Divergencija

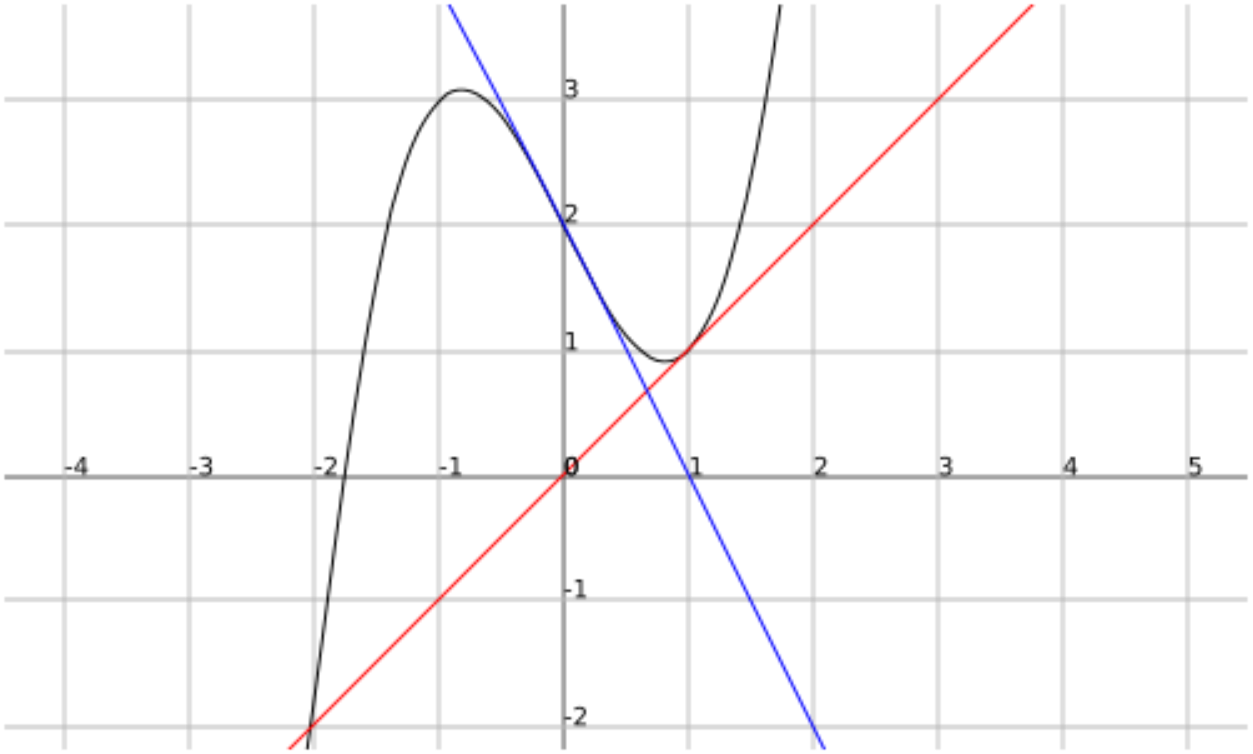
- Ako je početno rešenje daleko od nule, metoda divergira. Na primer:



Njutnova metoda Divergencija

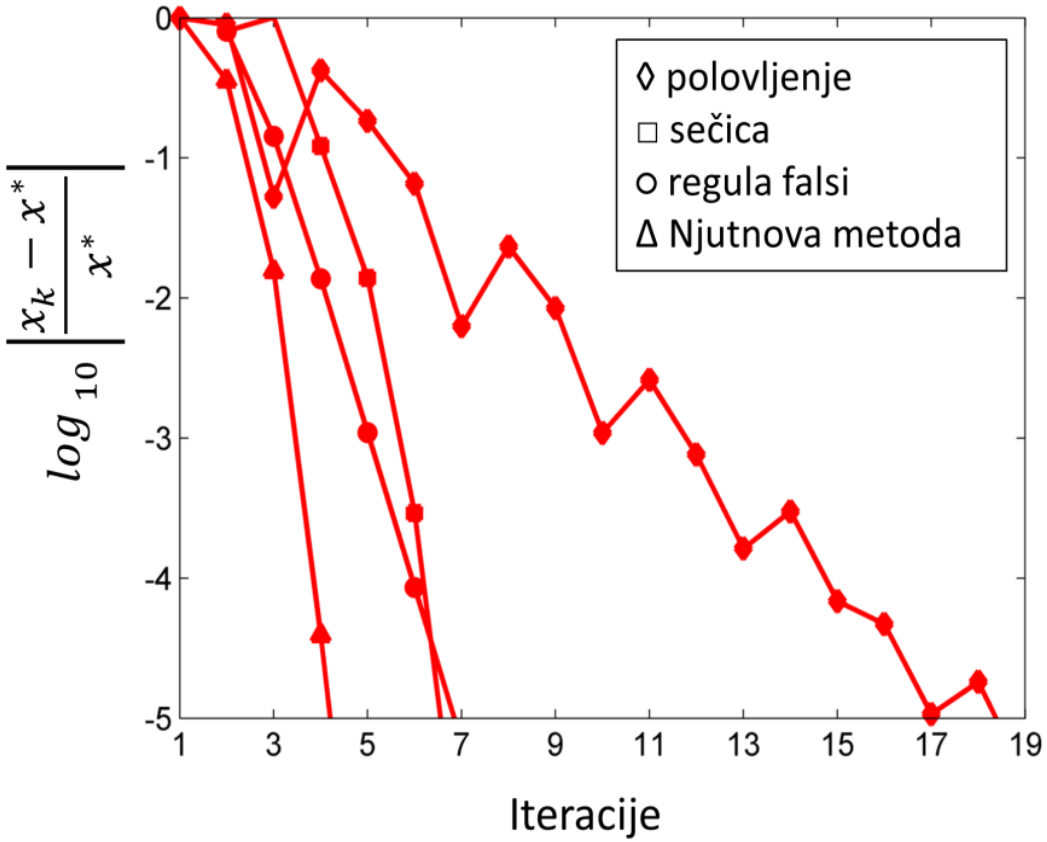
Funkcija je x^3-2x+2

Početno rešrnje je 0. Iteracije postupka beskonačno osciluju između 0 i 1.



Konvergencija metoda, Grafički prikaz

Ilustracija brzine konvergencije različitih numeričkih metoda za problem:
x-cos(x)=0



Rezime metoda

Metod	Prednosti	Mane
Polovljenje	-Garantovana konvergencija -Broj iteracija za toleranciju se može unapred odrediti -Laka za implementaciju -Ne zahteva prvi izvod	- Spora konvergencija - Zahteva da se rešenje nalazi u [a,b] tj. da važi $f(a)f(b)<0$
Sečica	-Brza konvergencija (sporija od Njutnove metode) -Ne zahteva prvi izvod	- Nema garantovanu konvergenciju - Zahteva dve početne tačke takve da važi $f(x_0)\neq f(x_1)$
Regula falsi	-Garantovana konvergencija (brža od polovljenja, sporija od sečice) -Ne zahteva prvi izvod	- Zahteva da se rešenje nalazi u $[x_0,x_1]$ tj. da važi $f(x_0)f(x_1)<0$
Njutnov	- Veoma brza konvergencija (ako je početno rešenje blizu nule)	- Nema garantovanu konvergenciju - Zahteva postojanje prvog izvoda i $f'(x_0)\neq 0$