Ekstremi funkcija jedne varijable

Pavković, Antonija

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Antonija Gerguri

EKSTREMI FUNKCIJA JEDNE VARIJABLE

ZAVRŠNI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Antonija Gerguri

Matični broj: Z-41521/12-Izv

Studij: Primjena informacijske tehnologije u poslovanju

EKSTREMI FUNKCIJA JEDNE VARIJABLE

ZAVRŠNI RAD

Mentor :

prof. Damir Horvat

Varaždin, rujan 2019.

Antonija Gerguri

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristila drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autorica potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

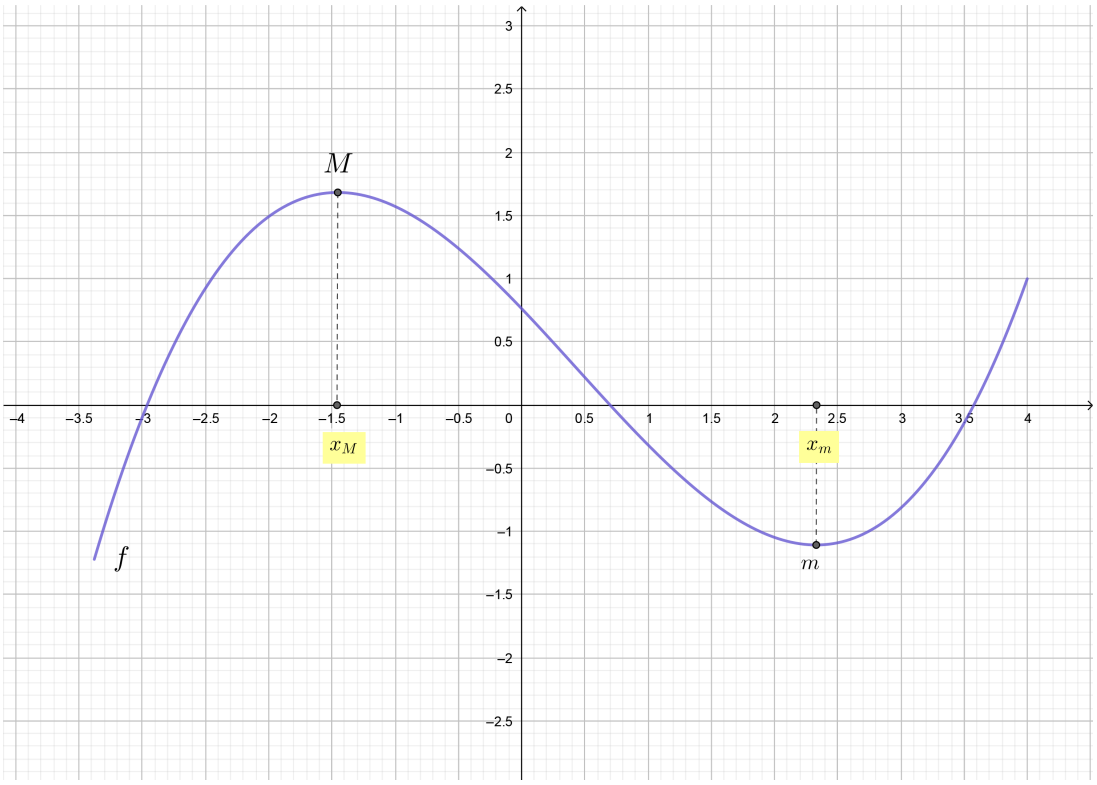
Sadržaj

1. Uvod

U ovom radu najprije će se objasniti pojam optimizacije i korak po korak doći će se do metoda koje se mogu koristiti kod rješavanja problema optimizacije za funkcije jedne varijable. Optimizacija je proces pronalaženja najboljeg rješenja za zadani problem [1]. Prema tome optimalno znači minimum ili maksimum funkcije. Definicije minimuma i maksimuma s prikazom grafova (slika 1) su sljedeće:

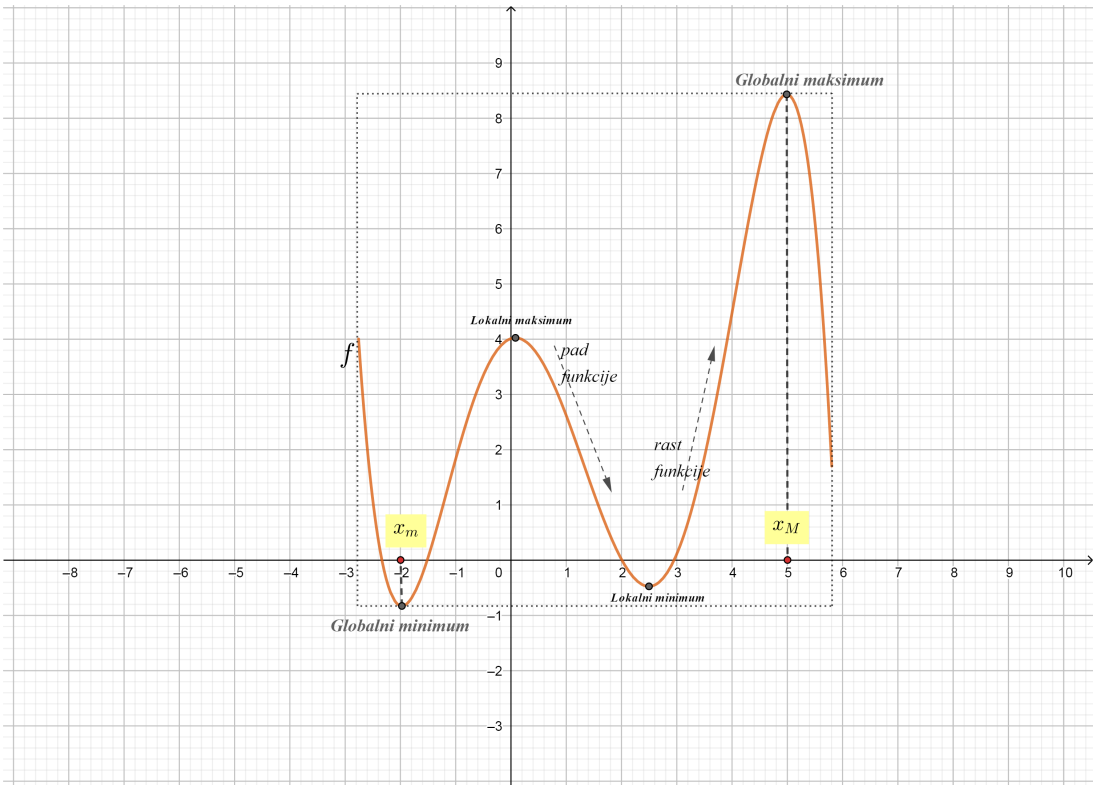
Definicija 1. *Kažemo da funkcija f ima lokalni maksimum u točki x_M ako unutar domene funkcije f postoji okolina O točke x_M takva da je na toj okolini $f(x_M)$ najveća vrijednost funkcije f, tj. $f(x) \leqslant f(x_M), \forall x \in O$. Ako vrijedi stroga nejednakost za sve $x \in O \setminus \{x_M\}$, tada govorimo o strogom lokalnom maksimumu.*

Definicija 2. *Kažemo da funkcija f ima lokalni minimum u točki x_m ako unutar domene funkcije f postoji okolina O točke x_m takva da je na toj okolini $f(x_m)$ najmanja vrijednost funkcije f, tj. $f(x) \geqslant f(x_m), \forall x \in O$. Ako vrijedi stroga nejednakost za sve $x \in O \setminus \{x_m\}$, tada govorimo o strogom lokalnom minimumu.*



Slika 1: Grafički prikaz lokalnog minimuma i maksimuma

Ako u nekoj točki funkcija ima lokalni maksimum, to ne znači da u toj točki funkcija poprima najveću vrijednost globalno, može se dogoditi da u nekim drugim točkama poprima veće vrijednosti, ali na nekoj dovoljno maloj okolini te točke to je najveća vrijednost. Dakle, riječ "lokalni" je ovdje jako bitna. Ista stvar je i za lokalni minimum. Na sljedećoj slici prikazat ćemo više lokalnih ekstrema s jednim globalnim minimumom i globalnim maksimumom (slika 2).



Slika 2: Grafički prikaz globalnog minimuma i maksimuma

Definicija 3. *Za funkciju $f : D_f \to \mathbb{R}, D_f \subseteq \mathbb{R}$, kažemo da raste ako vrijedi*

$$(\forall x_1, x_2 \in D_f)(x_1 < x_2 \Rightarrow f(x_1) \leqslant f(x_2)).$$

Definicija 4. *Za funkciju $f : D_f \to \mathbb{R}, D_f \subseteq \mathbb{R}$, kažemo da strogo raste ako vrijedi*

$$(\forall x_1, x_2 \in D_f)(x_1 < x_2 \Rightarrow f(x_1) < f(x_2)).$$

Definicija 5. *Za funkciju $f : D_f \to \mathbb{R}, D_f \subseteq \mathbb{R}$, kažemo da pada ako vrijedi*

$$(\forall x_1, x_2 \in D_f)(x_1 < x_2 \Rightarrow f(x_1) \geqslant f(x_2)).$$

Definicija 6. *Za funkciju $f : D_f \to \mathbb{R}, D_f \subseteq \mathbb{R}$, kažemo da strogo pada ako vrijedi*

$$(\forall x_1, x_2 \in D_f)(x_1 < x_2 \Rightarrow f(x_1) > f(x_2)).$$

U praksi cilj je imati informaciju o tome što se želi postići: maksimizirati dobit, učinkovitost ili pak minimizirati troškove ovisno o tome što se promatra. Zbog toga u slučaju nekog problema ili cilja koji ima više rješenja optimizacije pronalazi se najbolje rješenje problema u smislu nekog kriterija izvedbe. Postoji nekoliko općih pristupa kod rješavana problema optimizacije realne funkcije jedne varijable. Neke od tih pristupa objasnit će se u ovom radu.

Pristupi optimizacije:

1. Grafička metoda
2. Analitička metoda
3. Numerička metoda

Grafička metoda prikazuje funkciju u koordinatnom sustavu. Kao što sama riječ kaže to je vizualna metoda kojom tražimo optimalna rješenja za zadani problem odnosno funkciju. Ta rješenja predstavljaju minimum ili maksimum koje lako možemo uočiti na grafu. Ovakvu metodu možemo koristiti kod rješavanja manje kompleksnih problema te je najbrži način da saznamo za postojanje ekstrema. Gledajući sliku 1 vidimo da postoje ekstremi, ali ne znamo precizno koordinate što predstavlja nedostatak ove metode. Ako nam je potrebna takva informacija, trebamo koristiti analitičku ili neku numeričku metodu.

Analitička metoda se koristi kada su poznata analitička svojstva funkcije. Prvo se pronađe derivacija f' funkcije f i nakon toga se rješava jednadžba $f'(x) = 0$. Rješenja predstavljaju *stacionarne točke* koje su kandidati za *lokalne ekstreme* (minimum i maksimum). Također, ako je zadan interval $[a, b]$, tada se postižu *globalni ekstermi* i mogu se postizati u rubnim točkama segmenta. Prednosti ove metode su da dobivamo precizne koordinate ekstrema, ali ukoliko je jednadžba $f'(x) = 0$ komplicirana za riješiti tada koristimo numeričke metode.
Numeričke metode koriste iterativne postupke za generiranje niza

$$x_1, x_2 \ldots, x_k, \ldots .$$

Članovi tog niza se koriste kod traženja optimalne vrijednosti funkcije f. Postupak se prekida kada dođe do člana niza koji je dovoljno blizu optimalnog rješenja. Numeričke metode se koriste za rješavanje vrlo složenih optimizacijskih problema pomoću računala. Svaka od numeričkih metoda ima svoje prednosti i mane koje će se kasnije u radu napomenuti.

2. Analitička metoda i primjena derivacija

Kod analitičke metode najvažnija je primjena derivacija za rješavanje problema optimizacije. Problemi optimizacije javljaju se na primjer u ekonomiji kada se treba maksimizirati funkcija profita ili minimizirati funkcija troškova [2]. Također, promatra se rast i pad funkcije koje najlakše promatramo na grafu, a kod primjene u ekonomiji mislimo na profit koji raste ili pada s obzirom na količinu proizvodnje.

2.1. Derivacija funkcije

Pojam derivacije vezan je s problemom tangente. Intuitivno, tangenta je pravac koji dira graf funkcije f u zadanoj točki $T_0(x_0, f(x_0))$ pa se problem onda svodi na traženje koeficijenta smjera tangente. Nadalje, potrebna je još jedna točka $T_1(x_1, f(x_1))$ na grafu funkcije f kako bi se povukao pravac kroz te dvije točke, a taj pravac predstavlja sekantu. Prema tome koeficijent smjera k_s sekante s nije teško pronaći. Koeficijent smjera sekante dan je formulom

$$k_s = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{\Delta f}{\Delta x}.$$

Kada se točka T_1 postupno približava prema točki T_0 po grafu funkcije f, tada se sekanta sve više približava tangenti. Prema tome se zaključuje da Δx teži prema nuli pa tako objašnjavamo koeficijent smjera tangente prema formuli

$$k_t = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}.$$

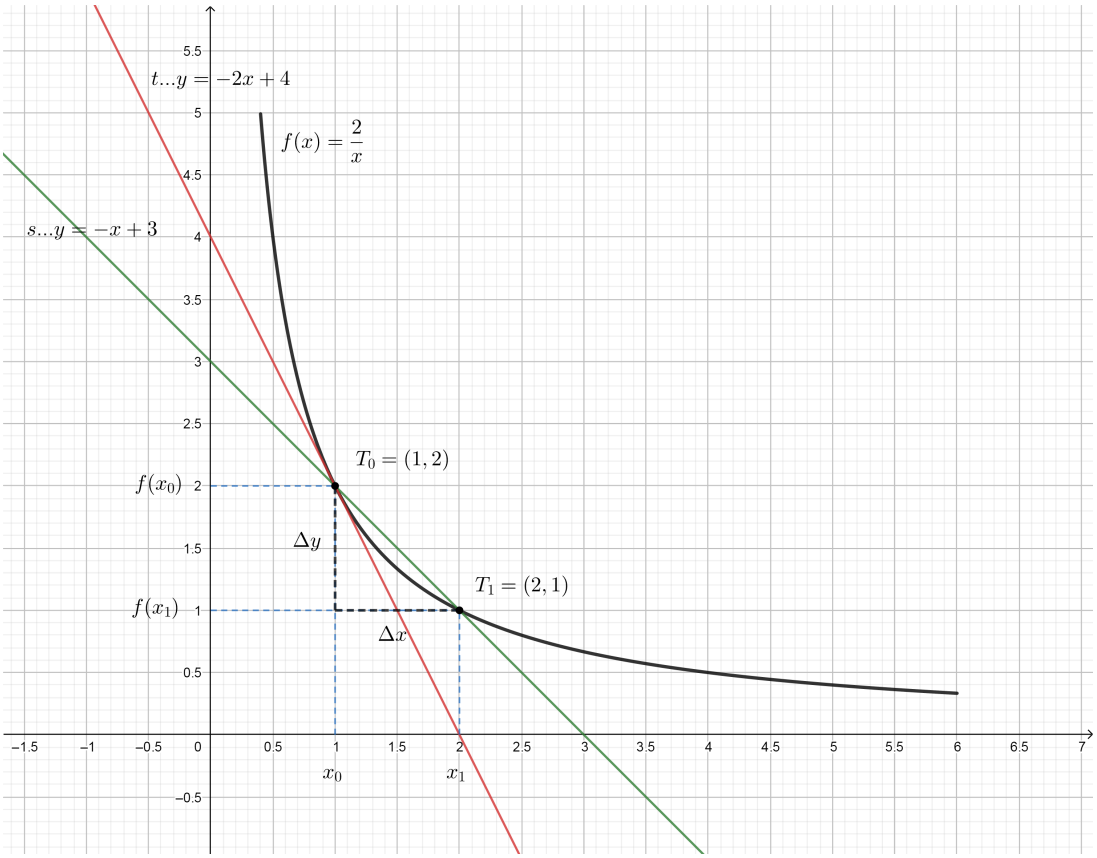
Ako navedeni limes postoji, derivacija funkcije f u točki x_0 je broj

$$f'(x_0) = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}.$$

Nakon što imamo koeficijent smjera tangente k_t, jednadžbu tangente t možemo naći kao jednadžbu pravca koja prolazi kroz točku $T_0(x_0, y_0)$:

$$t \ldots y - y_0 = k_t(x - x_0).$$

Na sljedećem primjeru (slika 3) prikazana je tangenta u točki $T_0(1, 2)$ na grafu funkcije $f(x) = \frac{2}{x}$. Također, prikazana je sekanta kroz točke $T_0(1, 2)$ i $T_1(2, 1)$.



Slika 3: Pravac sekante i tangente za zadanu funkciju

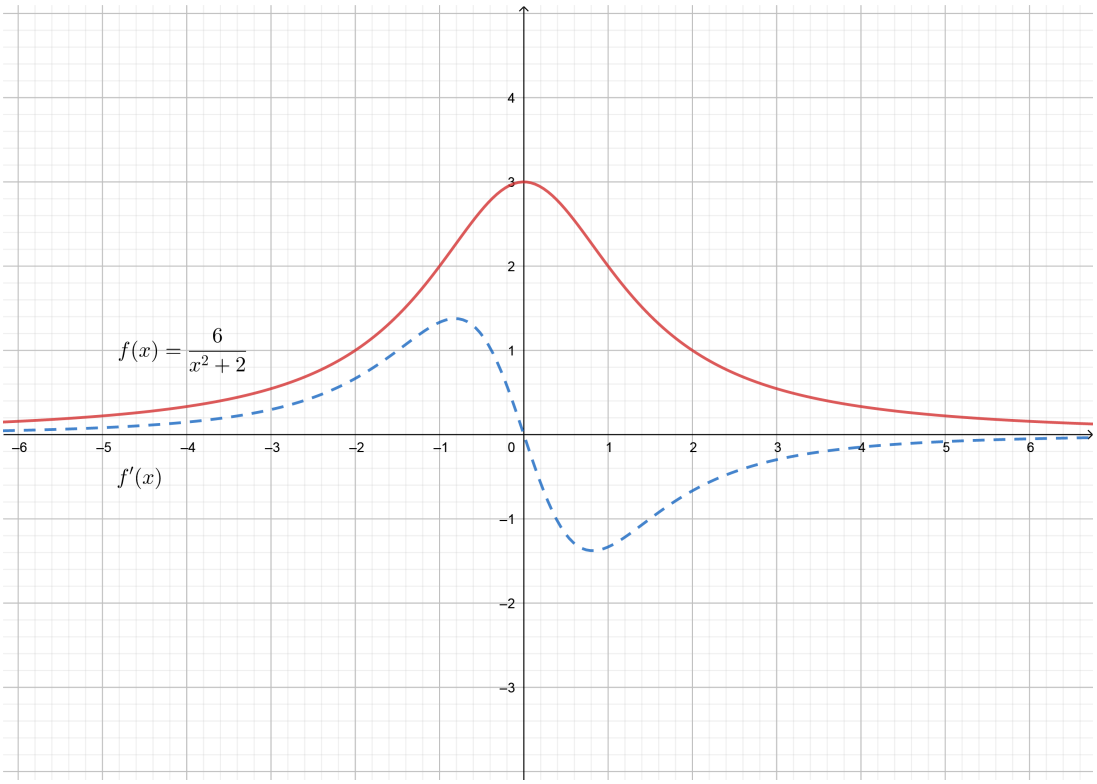
2.2. Ekstremi funkcije i monotonost

U ekonomiji se često postavlja pitanje raste li profit ili pada s obzirom na količinu proizvodnje. To pitanje u matematici povezujemo s monotonosti odnosno s rastom ili padom funkcije zbog čega su nam potrebne derivacije za rješenje tog problema. Kao što možemo vidjeti na slici 4, vrijede sljedeći teoremi.

Teorem 1. *Diferencijabilna funkcija* $f : \langle a, b\rangle \to \mathbb{R}$ *raste na intervalu* $\langle a, b\rangle$ *akko je* $f'(x) \geqslant 0$ *za svaki* $x \in \langle a, b\rangle$*.*

Teorem 2. *Diferencijabilna funkcija* $f : \langle a, b\rangle \to \mathbb{R}$ *pada na intervalu* $\langle a, b\rangle$ *akko je* $f'(x) \leqslant 0$ *za svaki* $x \in \langle a, b\rangle$*.*

Navedena zapažanja vrijede i općenito.



Slika 4: Graf funkcije i njena derivacija

Za diferencijabilne strogo monotone funkcije vrijede teoremi:

Teorem 3. *Diferencijabilna funkcija* $f : \langle a, b\rangle \to \mathbb{R}$ *strogo raste na intervalu* $\langle a, b\rangle$ *akko je* $f'(x) \geqslant 0$ *za svaki* $x \in \langle a, b\rangle$ *i skup*

$$\{x \in \langle a, b\rangle : f'(x) = 0\}$$

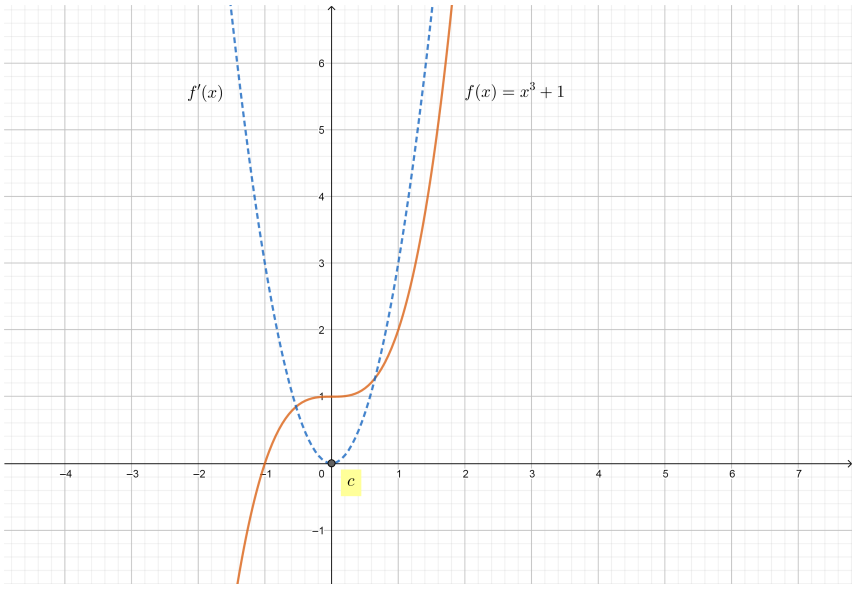
ne sadrži interval.

Teorem 4. *Diferencijabilna funkcija* $f : \langle a, b\rangle \to \mathbb{R}$ *pada na intervalu* $\langle a, b\rangle$ *akko je* $f'(x) \leqslant 0$ *za svaki* $x \in \langle a, b\rangle$ *i skup*

$$\{x \in \langle a, b\rangle : f'(x) = 0\}$$

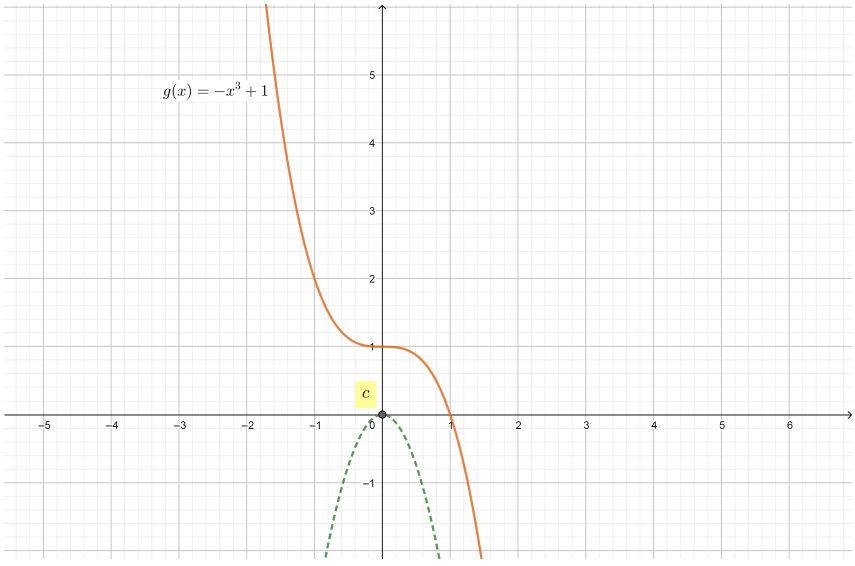
ne sadrži interval.

Strogo monotone diferencijabilne funkcije mogu u nekim slučajevima imati derivaciju jednaku 0. Primjer su funkcije $f(x) = x^3 + 1$ i $g(x) = -x^3 + 1$ na slikama 5 i 6.



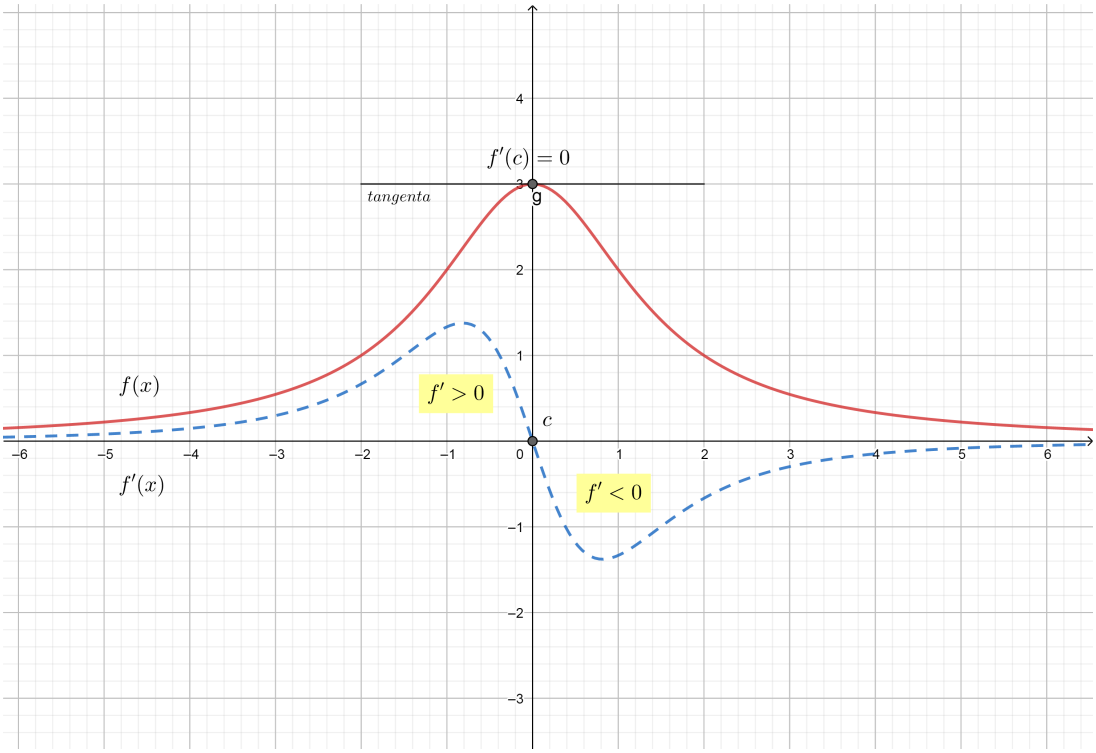
Slika 5: Strogo rastuća funkcija

Iz slika 5 i 6 zaključuje se da strogo monotone funkcije imaju derivaciju jednaku 0 samo na konačnom skupu, a ako je je taj skup beskonačan njegovi elementi moraju se moći poredati u niz realnih brojeva.



Slika 6: Strogo padajuća funkcija

2.2.1. Stacionirane točke



Slika 7: Stacionirana točka koja je ujedno i ekstrem

U točkama lokalnih ekstrema tangenta je paralelana s osi x pa je i njezin koeficijent smjera k_t jednak 0. Iz toga vrijedi Fermatov terorem [2] da je i derivacija funkcije u točki lokalnog ekstrema jednaka 0.

Teorem 5. *Neka je $f : \langle a, b\rangle \to \mathbb{R}$ funkcija koja u točki c ima lokalni ekstrem. Ako je f diferencijabilna u točki c, tada je $f'(c) = 0$.*

Za točke koje imaju lokalni ekstrem, ali ne i derivaciju, ne možemo primjeniti Fermatov teorem. Nadalje, uvjet $f'(c) = 0$ je samo nužan za diferencijabilne funkcije (slika 5 i 6), ali ne i dovoljan da bi funkcija u toj točki c imala lokalni ekstrem. Zbog toga, to je još jedan razlog zašto nekad treba koristiti numeričke metode za traženje ekstrema koje ne koriste derivaciju funkcije.

Definicija 7. *Stacionarne točke funkcije f su točke u kojima je derivacija funkcije f jednaka 0. Drugim riječima, to su rješenja jednadžbe $f'(x) = 0$.*

Točka u kojoj je derivacija jednaka 0 može i ne mora biti ekstrem. Zato kažemo da su stacionarne točke kandidati za ekstreme diferencijabilne funkcije. Primjeri u kojima je derivacija jednaka 0 i funkcija nema ekstreme su na slikama 5 i 6.

Za ispitivanje stacionarnih točaka koristimo prvu i drugu derivaciju što ćemo objasniti u slijedećim poglavljima.

2.2.2. Ispitivanje stacionarnih točaka pomoću prve derivacije

Prvo se za zadanu funkciju f odredi derivacija te se nakon toga traže stacionarne točke. Stacionarne točke određuju se rješavanjem jednandžbe $f'(x) = 0$. Nakon toga provjeravamo postižu li se lokalni ekstremi u tim točkama. Kao što možemo vidjeti na slici 7 postiže se lokalni ekstrem u točki c, gdje se razlikuje predznak prve derivacije f' s lijeve i desne strane. Gledajući slike 5 i 6 ne postižu se lokalni ekstremi jer se predznaci prve derivacije f' ne razlikuju s lijeve i desne strane u točki c. Navedeno opažanje vrijedi i općenito.

Teorem 6. *Stacionarna točka c diferencijabilne funkcije f je lokalni ekstrem ako i samo ako postoji okolina točke c takva da se na toj okolini predznak prve derivacije lijevo od točke c razlikuje od predznaka prve derivacije desno od točke c.*

Primjer 1. Odredimo lokalne ekstreme funkcije $f(x) = -\frac{2}{3}x^3 - \frac{3}{2}x^2 + 2x + 1$.

Derivacija funkcije f jednaka je

$$f'(x) = 2x^2 - 3x + 2.$$

Rješavanjem jednadžbe $f'(x) = 0$ određujemo stacionarne točke:

$$2x^2 - 3x + 2 = 0.$$

Rješenja kvadratne jednadžbe su $x_1 = -2$ i $x_2 = -\frac{1}{2}$.

Dalje treba ispitati karakter stacionarnih točaka, tj. postiže li funkcija lokalne ekstreme u nekima od njih. Ispitivanje pomoću prve derivacije obavlja se pregledno preko tablice. U tablici naznačimo domenu promatrane funkcije i unutar domene redom po veličini smjestimo sve stacionarne točke.

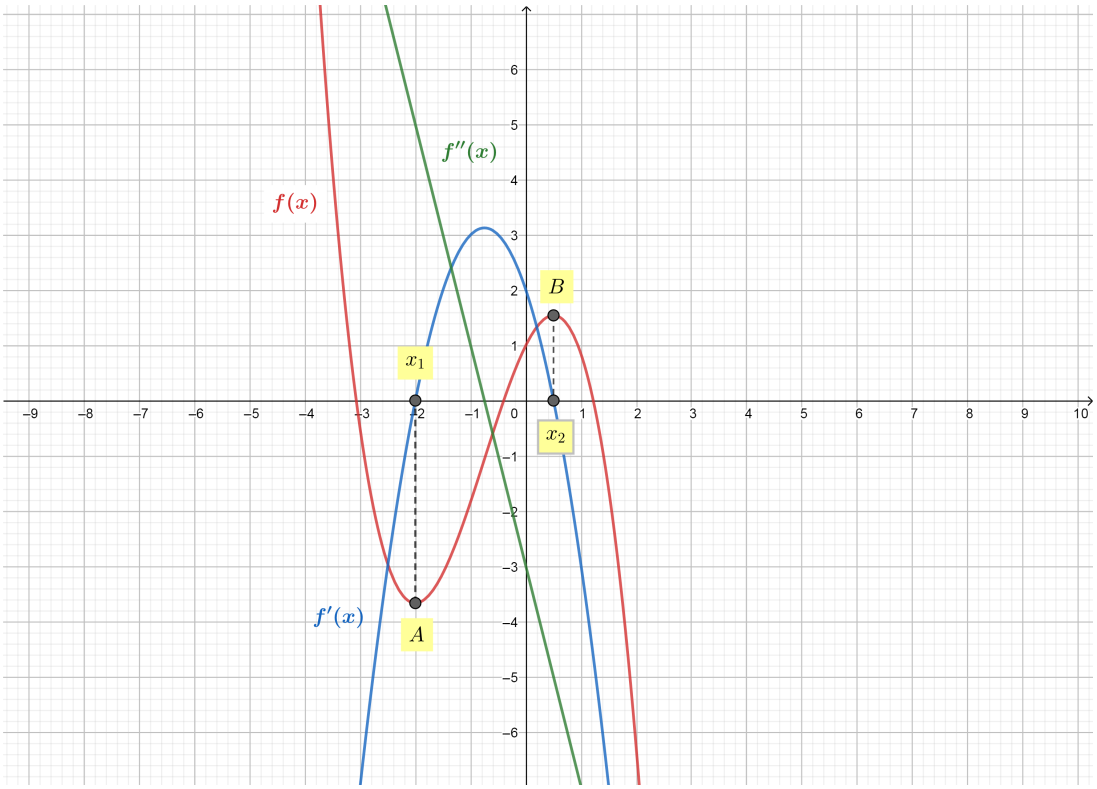
Kako je $D_f = \mathbb{R}$, tj. $D_f = \langle -\infty, +\infty \rangle$, a stacionarne točke su -2 i $\frac{1}{2}$, tablica u ovom slučaju izgleda ovako:

	$-\infty$		-2		$\frac{1}{2}$		$+\infty$
f'		$-$		$+$		$-$	
f		$\searrow$		$\nearrow$		$\searrow$	

Tablica 1: Prikaz monotonosti zadane funkcije

Da bismo popunili tablicu na ovaj način, na svakom pojedinom intervalu odaberemo neku njegovu unutarnju točku i izračunamo vrijednost derivacije u toj točki.

- Ako je derivacija pozitivna, tada na tom intervalu funkcija raste, pišemo znak $+$ i znak $\nearrow$.
- Ako je derivacija negativna, tada na tom intervalu funkcija pada, pišemo znak $-$ i znak $\searrow$.



Slika 8: Primjer funkcije, njezine derivacije i ekstremi te funkcije

Pogledom na tablicu 1 i sliku 8 možemo zaključiti da funkcija postiže lokalne ekstreme u obje točke jer se u okolinama obje stacionarne točke predznaci derivacije razlikuju.

2.2.3. Ispitivanje stacionarnih točaka pomoću druge derivacije

Ono što primjećujemo na slici 8 je da na području točke A, odnosno lokalnog minimuma, funkcija f'' je veća od nule. Dok na području točke B, odnosno lokalnog maksimuma, funkcija f'' je manja od nule.

Teorem 7. *Ako je $f'(c) = 0$ i $f''(c) > 0$, onda funkcija f u točki c ima lokalni minimum. Ako je $f'(c) = 0$ i $f''(c) < 0$, onda funkcija f u točki c ima lokalni maksimum. Ako je $f'(c) = 0$ i $f''(c) = 0$, onda ovaj test ne daje odgovor pa ispitivanje moramo obaviti pomoću nekog drugog testa.*

Primjer 2. Odredimo lokalne ekstreme funkcije $f(x) = -\frac{2}{3}x^3 - \frac{3}{2}x^2 + 2x + 1$ iz predhodnog primjera 1 pomoću druge derivacije.

Druga derivacija funkcije f jednaka je

$$f''(x) = -4x - 3.$$

Uvrštavamo stacionarne točke u funkciju $f''(x)$.

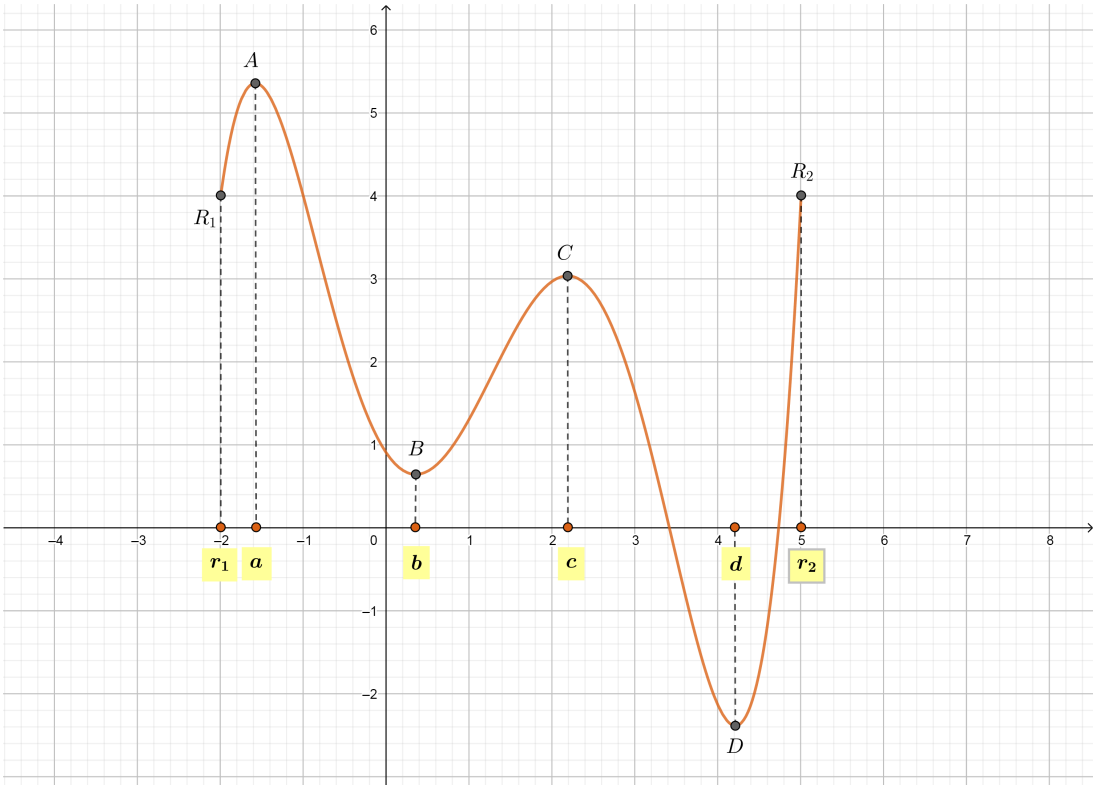
$$f''(-2) = -4 \cdot (-2) - 3 = 5$$
$$f''\left(\frac{1}{2}\right) = -4 \cdot \frac{1}{2} - 3 = -5$$

Dobiveni rezultati dovode do sljedećih zaključaka:

- $f''(-2) = 5 > 0$, funkcija f u točki -2 ima lokalni minimum koji iznosi $f(-2) = -\frac{11}{3}$.
- $f''(\frac{1}{2}) = -5 < 0$, funkcija f u točki -5 ima lokalni maksimum koji iznosi $f(\frac{1}{2}) = \frac{37}{24}$.

2.3. Ekstremi funkcije na segmentu

Svaka neprekidna funkcija na segmentu $[a, b]$ poprima globalni minimum i globalni maksimum. Za intervale oblika $\langle a, b]$, $[a, b\rangle$ i $\langle a, b\rangle$ to više ne mora biti slučaj. Na takvim intervalima neprekidna funkcija može, ali i ne mora poprimati globalne ekstreme ili može poprimati samo neki od globalnih ekstrema.



Slika 9: Primjer funkcije te globalni i lokalni ekstremi

Ako postavimo različite segmente na sliku 9 možemo dobiti različite globalne i lokalne ekstreme:

- Na segmetu $[r_1, r_2]$ funkcija postiže globalni maksimum u točki a, globalni minimum u točki d, lokalne maksimume u točkama c i r_2 te lokalne minimume u točkama r_1 i b. Na intervalu $\langle r_1, r_2\rangle$ funkcija postiže iste globalne ekstreme, ali isključuje lokalne ekstreme u točkama r_1 i r_2.
- Na segmentu $[r_1, b]$ funkcija postiže globalni minimum u točki b i globalni maksimum u točki a te nam ostaje lokalni minimum u točki r_1. Na intervalu $[r_1, b\rangle$ funkcija postiže lokalni minimum u točki r_1 i globalni maksimum u točki a, a na intervalu $\langle r_1, b\rangle$ funkcija postiže samo globalni maksimum u točki a.
- Na segmentu $[b, c]$ funkcija postiže globalni maksimum i minimum, na intervalu $[b, c\rangle$ funkcija postiže samo globalni minimum, dok na intervalu $\langle b, c]$ postiže samo globalni maksimum, dok na intervalu $\langle b, c\rangle$ funkcija nema globalnih ekstrema.

Ako imamo zadanu funkciju f koja je neprekidna na segmentu $[a, b]$, znamo da ta funkcija ima globalni minimum i globalni maksimum na segmentu $[a, b]$.
Drugim riječima, postoje $m, M \in \mathbb{R}$ takvi da je $m \leqslant f(x) \leqslant M$ za svaki $x \in [a, b]$. Zapravo vrijedi i više, tj. Im $f = [m, M]$. Iz toga slijedi da neprekidna funkcija na segmentu poprima sve međuvrijednosti između globalnog minimuma m i globalnog maksimuma M.

3. Numeričke metode za traženje ekstrema

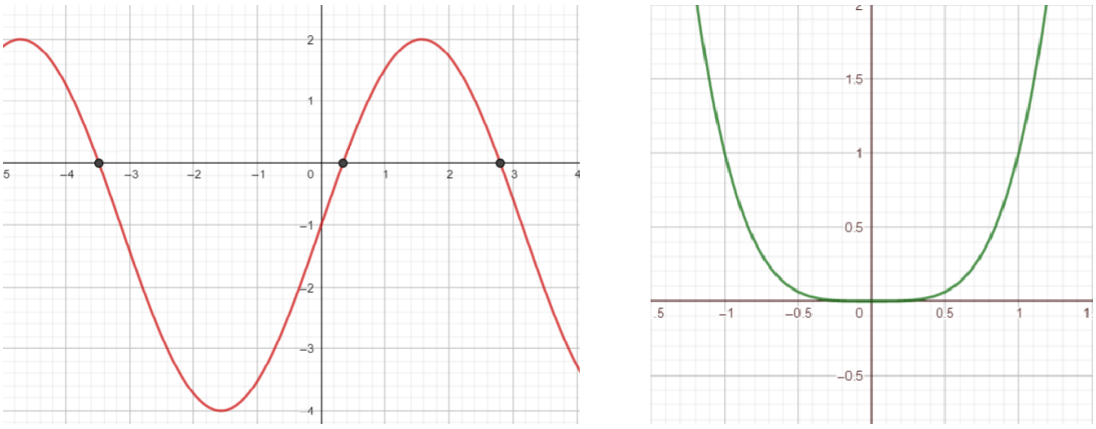
U praksi uglavnom nije moguće egzaktno riješiti jednadžbu $f'(x) = 0$. Zbog toga je potrebno primijeniti neke numeričke metode za traženje nultočaka i time se traženje ekstrema ne svodi direktno na funkciju f, već na njenu derivaciju f'. To su onda takozvane indirektne metode za traženje ekstrema koje osim zadane funkcije f koriste i njezine derivacije (prvu, drugu,... ovisno o metodi).
Ako funkcija nije derivabilna ili je derivacija iz nekog razloga prekomplicirana, tada se mogu koristiti direktne metode pretraživanja koje za traženje ekstrema koriste samo informaciju o funkciji f prilikom generiranja niza $x_1, x_2, \ldots, x_k, \ldots$
U ovom dijelu rada prikazat će se različite metode za rješavanje tih problema. Za svaki problem postoji metoda koja je učinkovitija za određeni problem od drugih metoda i takve metode lako možemo implementirati na računalu.
Cilj numeričkih metoda je iterativnim postupkom reducirati početno zadani interval, a svaka numerička metoda ima svoje postupke za dobivanje zadovoljavajuće aproksimacije egzaktnog rješenja. Postoje dvije opće klase jednodimenzionalne optimizacije:

1. Indirektne metode – osim zadane funkcije koriste i njezine derivacije
2. Direktne metode pretraživanja – koriste samo zadanu funkciju

Svaka od ovih metoda zahtijeva segment I unutar kojeg se nalazi nultočka.
Nedostatak indirektnih metoda jest da osim funkcije f trebaju i njezine derivacije, a prednost je da su brže. S druge strane, brže postizanje rezultata može utjecati na sigurnu konvergenciju tih metoda. Prednost direktnih metoda jest što ne trebaju derivaciju funkcije pa se mogu primijeniti na šire klase funkcija, ali su sporije od indirektnih metoda.

3.1. Indirektne metode

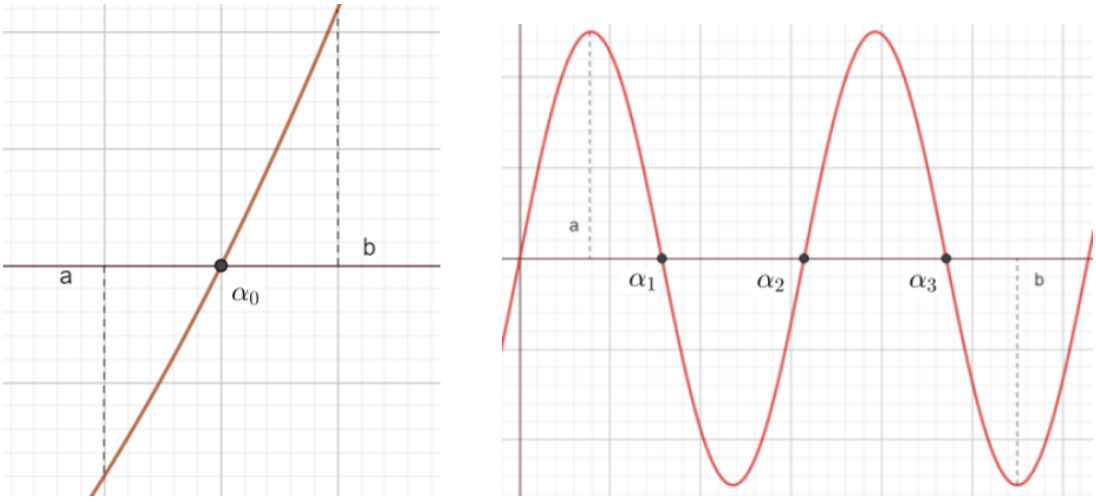
Indirektne metode koriste se za određivanje nultočaka koje su potrebne kako bi se došlo do ekstrema funkcije f. Kod određivanja nultočke potrebni su iterativni postupci koji zahtijevaju procjenu početnog intervala. Ako je jednadžba povezana s fizičkim problemom, predložit će se približna lokacija nultočke, a drugi način je provođenje numeričkog postupka. Procjena početnog intervala je jako bitna jer u protivnom se može izgubiti konvergencija ili će se konvergencija postići na nultočki koja nije tražena. Nadalje, funkcija f mora biti neprekidna na segmentu I i mora imati izolirane nultočke (inače se javlja problem konvergencije).



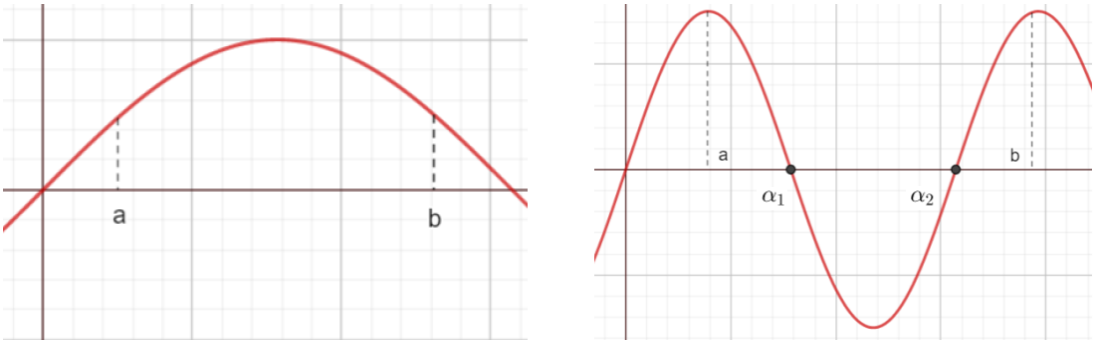
Slika 10: Prikaz izoliranih (lijevo) i neizoliranih (desno) nultočaka

Indirektne metode se razlikuju po brzini konvergencije i sigurnoj konvergenciji. Sporije metode imaju sigurnu konvergenciju, dok kod bržih može doći do udaljavanja od rješenja ako nisu zadovoljeni dodatni uvjeti specifični za pojedinu metodu.

Svako rješenje jednadžbe $f(x) = 0$ predstavlja nultočku funkcije f. Na intervalu $I = [a, b]$ funkcija f može imati jednu, nijednu ili više nultočaka.



Slika 11: Uvjet $f(a)f(b) < 0$ i postojanje nultočaka



Slika 12: Uvjet $f(a)f(b) > 0$ i postojanje nultočaka

Neprekidna funkcija f na segmentu $[a, b]$ ima barem jednu nultočku ako u rubnim točkama segmenta postiže vrijednosti koje imaju suprotne predznake, tj. ako vrijedi $f(a)f(b) < 0$ (slika 11). To je dovoljan uvjet, ali ne i nužan (slika 12).

Isti predznak rubnih vrijednosti na intervalu može uzrokovati postojanje jedne, više ili pak niti jedne nultočke (slika 12).

Postoje dva općenita kriterija zaustavljanja iterativnog postupka kada se postiže određena točnost:

1. Dinamička ocjena greške koristi se za dobivanje udaljenosti aproksimacije nultočke od prave nultočke. Uvjeti koje funkcija treba ispuniti su da mora imati neprekidnu derivaciju te derivacija treba biti konstantnog predznaka na segmentu $[a, b]$ tj. $m_1 > 0$. U suprotnom, ne može se koristiti ovaj kriterij osim ako se metodom bisekcije ne smanji segment $[a, b]$, sve dok se ne postigne $m_1 > 0$. Sve metode, osim metode bisekcije, pokreću se na intervalu na kojemu je funkcija strogo monotona i nema točaka infleksije, kako bi se izbjegla pojava poteškoća. Nedostatak ovog kriterija jest što zahtijeva računanje vrijednosti m_1 i poznavanje derivacije funkcije f.

   Kako bi se postiglo $|\alpha - x_n| < \varepsilon$ dovoljno je zahtijevati

   $$|f(x_n)| < m_1\varepsilon, \quad m_1 = \min_{x\in[a,b]} |f'(x)|.$$

2. Ako se teško pronalazi m_1 ili je $m_1 = 0$, koristi se drugi kriterij koji uz unaprijed zadane željene točnosti ε i δ zahtijeva provjeravanje dviju nejednakosti

   $$|f(x_n)| < \varepsilon, \quad |x_n - x_{n-1}| < \delta.$$

   Prednost ovog kriterija jest što koristi samo funkciju f, ali mora pamtiti prethodnu aproksimaciju nultočke.

Kriteriji se trebaju prilagoditi traženju ekstrema tako da se funkcija f zamijeni s njenom derivacijom f' koja će predstavljati početnu funkciju, a derivacija funkcije f' zamijenit će se s drugom derivacijom f'' jer ona predstavlja derivaciju početne funkcije f'.
Iz toga, prvi kriterij zaustavljanja glasi

$$|f'(x_n)| < m_1\varepsilon, \quad m_1 = \min_{x\in[a,b]} |f''(x)|,$$

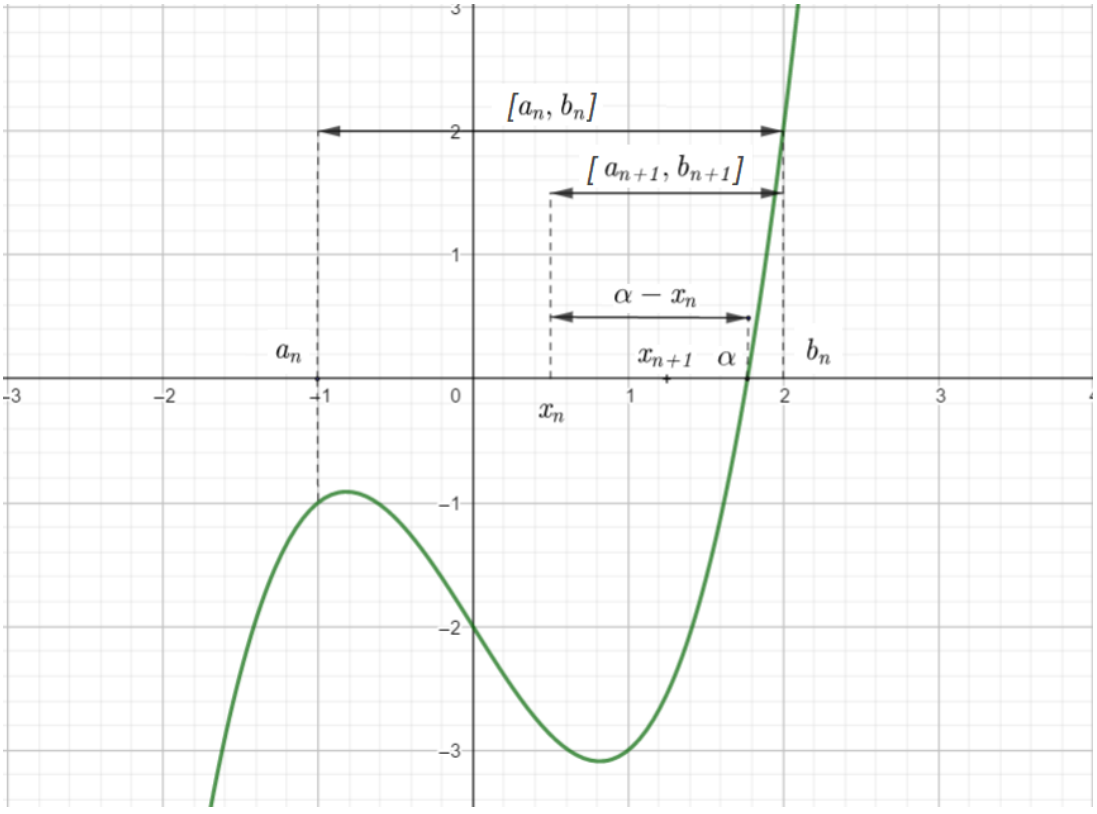
a drugi kriterij zaustavljanja glasi

$$|f'(x_n)| < \varepsilon, \quad |x_n - x_{n-1}| < \delta.$$

3.1.1. Metoda bisekcije

Metoda bisekcije predstavlja zatvorenu metodu i zaustavlja se kada segment postane dovoljno reduciran evaluiranjem iterativnog postupka. Svakom iteracijom novi segment $[a_{n+1}, b_{n+1}]$ je dvostruko manje duljine od prethodnog segmenta $[a_n, b_n]$. Stoga se ova metoda naziva metodom raspolavljanja. Najveća prednost ove metode jest da ne koristi funkciju f, već samo duljinu trenutnog segmenta zbog čega nije najbrža dostupna metoda, ali je najpouzdanija. Koraci za primjenu:

1. Funkcija f treba biti neprekidna na segmentu $[a, b]$ i mora zadovoljavati uvjet $f(a)f(b) < 0$.
2. Raspolavlja se postavljeni segment i ako polovište nije nultočka usredotoči se na onu polovicu segmenta u čijim rubovima funkcija f ima različite predznake.
3. Ponavlja se postupak na odabranu polovicu.
4. U trenutku kada je zadovoljen kriterij zaustavljanja, prekida se iteracija.



Slika 13: Metoda bisekcije

U prethodnom poglavlju spomenulo se da funkcija treba imati izolirane nultočke kako bi se lakše postigao uvjet $f(a)f(b) < 0$, inače neće biti moguće koristiti metodu bisekcije.
Konstrukcija segmenta $[a_n, b_n]$ sastoji se u raspolavljanju segmenta $[a_{n-1}, b_{n-1}]$ točkom x_{n-1} i to tako da je

$$\begin{aligned} a_n = x_{n-1}, b_n = b_{n-1} \quad &\text{ako je} \quad f(a_{n-1})f(x_{n-1}) > 0, \\ a_n = a_{n-1}, b_n = x_{n-1} \quad &\text{ako je} \quad f(a_{n-1})f(x_{n-1}) < 0. \end{aligned} \tag{♠}$$

Prema slici 13 zadani segment smanjuje se nizom aproksimacija (x_n) korištenjem formule

$$x_n = \frac{1}{2} \cdot (a_n + b_n)$$

pri čemu niz (x_n) konvergira prema nekom rješenju jednadžbe $f(x) = 0$ do određene točnosti ε.

Kriterij zaustavljanja specifičan samo za metodu bisekcije je

$$|b_n - x_n| < \varepsilon,$$

gdje je x_n polovište segmenta $[a_n, b_n]$. Kada se ispuni uvjet s unaprijed određenom točnošću ε, zaustavlja se iteracija i rješenje se nalazi unutar reduciranog segmenta.

Tema ovog rada je traženje ekstrema, a metoda bisekcije se koristi za traženje nultočaka. U tom slučaju potrebna je derivacija funkcije koja će se koristiti kao početna funkcija f', a njezine nultočke će predstavljati stacionarne točke funkcije f.

Primjer 3. S točnošću $\varepsilon = 0.001$ korištenjem metode bisekcije treba odrediti nultočku funkcije $f(x) = x^3 + 3x - 5$.

1. Treba pronaći segment na kojemu funkcija f ima barem jednu nultočku.

   $$f(0) = -5 < 0$$
   $$f(1) = -1 < 0$$
   $$f(2) = 9 > 0$$

   Nultočka sigurno postoji na segmentu na kojemu funkcija f u rubnim točkama ima vrijednosti suprotnog predznaka. Iz gore navedenog računa može se zaključiti da se to pojavljuje na segmentu $[1, 2]$.

2. Započinju iteracije $x_n = \frac{a_n + b_n}{2}$, a prije svake se provjerava uvjet $|b_n - x_n| < \varepsilon$.

3. Kreira se novi segment prema (♠).

4. Iterativni postupak se zaustavlja nakon što je ispunjen kriterij zaustavljanja $|b_n - x_n| < \varepsilon$.

```
>> [x, itCount] = bisect ( @funcx, 1, 2, 0.001)
a=1, fa=-1, x=1.5, fx=2.875, b=2, fb=9
a=1, fa=-1, x=1.25, fx=0.703125, b=1.5, fb=2.875
a=1, fa=-1, x=1.125, fx=-0.201171875, b=1.25, fb=0.703125
a=1.125, fa=-0.201171875, x=1.1875, fx=0.2370605469, b=1.25, fb=0.703125
a=1.125, fa=-0.201171875, x=1.15625, fx=0.01455688477, b=1.1875, fb=0.2370605469
a=1.125, fa=-0.201171875, x=1.140625, fx=-0.09414291382, b=1.15625, fb=0.01455688477
a=1.140625, fa=-0.09414291382, x=1.1484375, fx=-0.04000329971, b=1.15625, fb=0.01455688477
a=1.1484375, fa=-0.04000329971, x=1.15234375, fx=-0.01277595758, b=1.15625, fb=0.01455688477
a=1.15234375, fa=-0.01277595758, x=1.154296875, fx=0.0008772537112, b=1.15625, fb=0.01455688477
a=1.15234375, fa=-0.01277595758, x=1.153320312, fx=-0.005952651612, b=1.154296875, fb=0.0008772537112

broj iteracija = 10
x =  1.153320312500000
itCount =  10
```

Slika 14: Metoda bisekcije za funkciju $f(x) = x^3 + 3x - 5$ u Octave-u

n	a	b	$f(a)$	$f(b)$	x	$f(x)$	ε
0	1	2	-1	9	1.5	2.875	0.5
1	1	1.5	-1	2.875	1.25	0.703	0.25
2	1	1.25	-1	0.703	1.125	-0.201	0.125
3	1.125	1.25	-0.201	0.703	1.1875	0.237	0.0625
4	1.125	1.1875	-0.201	0.237	1.15625	0.0146	0.03125
5	1.125	1.1562	-0.201	0.146	1.1406	-0.094	0.0156
6	1.1406	1.1562	-0.094	0.146	1.1484	-0.04	0.0078
7	1.1484	1.1562	-0.040	0.146	1.1523	-0.0128	0.0039
8	1.1523	1.1562	-0.0128	0.146	1.1543	0.00088	0.0019
9	1.1523	1.15643	-0.00012776	0.00087725	1.1533	-0.0059527	0.001

Tablica 2: Metoda bisekcije za funkciju $f(x) = x^3 + 3x - 5$

U tablici 2 prikazane su zaokružene vrijednosti iteracija, dok su na slici 14 vidljive vrijednosti zaokružene na 10 decimala.
Vrijednost nultočke dobivene na traženu točnost od 0.001 je $x_{10} = 1.153320$.

Primjer 4. S točnošću $\varepsilon = 0.001$ korištenjem metode bisekcije treba odrediti ekstreme funkcije $f(x) = x^3 + x^2 - 3x$.

Za pronalazak ekstrema potrebna je derivacija zadane funkcije f i tada početna funkcija glasi

$$f'(x) = 3x^2 + 2x - 3.$$

Ponavljaju se koraci iz prethodnog primjera. Na početku se traži segment u čijim rubnim točkama f' poprima vrijednosti suprotnih predznaka.

$$f'(0) = -3 < 0$$
$$f'(1) = 2 > 0$$

Dobivamo segment $[0, 1]$ u kojem se nalazi nultočka derivacije f', odnosno u ovom slučaju ekstrem funkcije f. Nakon što smo pronašli početni segment, možemo početi s iterativnim postupkom.

```
>> [x, itCount] = bisect ( @funcx1, 0, 1, 0.001)
a=0, fa=-3, x=0.5, fx=-1.25, b=1, fb=2
a=0.5, fa=-1.25, x=0.75, fx=0.1875, b=1, fb=2
a=0.5, fa=-1.25, x=0.625, fx=-0.578125, b=0.75, fb=0.1875
a=0.625, fa=-0.578125, x=0.6875, fx=-0.20703125, b=0.75, fb=0.1875
a=0.6875, fa=-0.20703125, x=0.71875, fx=-0.0126953125, b=0.75, fb=0.1875
a=0.71875, fa=-0.0126953125, x=0.734375, fx=0.08666992188, b=0.75, fb=0.1875
a=0.71875, fa=-0.0126953125, x=0.7265625, fx=0.03680419922, b=0.734375, fb=0.08666992188
a=0.71875, fa=-0.0126953125, x=0.72265625, fx=0.01200866699, b=0.7265625, fb=0.03680419922
a=0.71875, fa=-0.0126953125, x=0.720703125, fx=-0.0003547668457, b=0.72265625, fb=0.01200866699
a=0.720703125, fa=-0.0003547668457, x=0.7216796875, fx=0.00582408905, b=0.72265625, fb=0.01200866699

broj iteracija = 10
x =    7.216796875000000e-01
itCount =  10
```

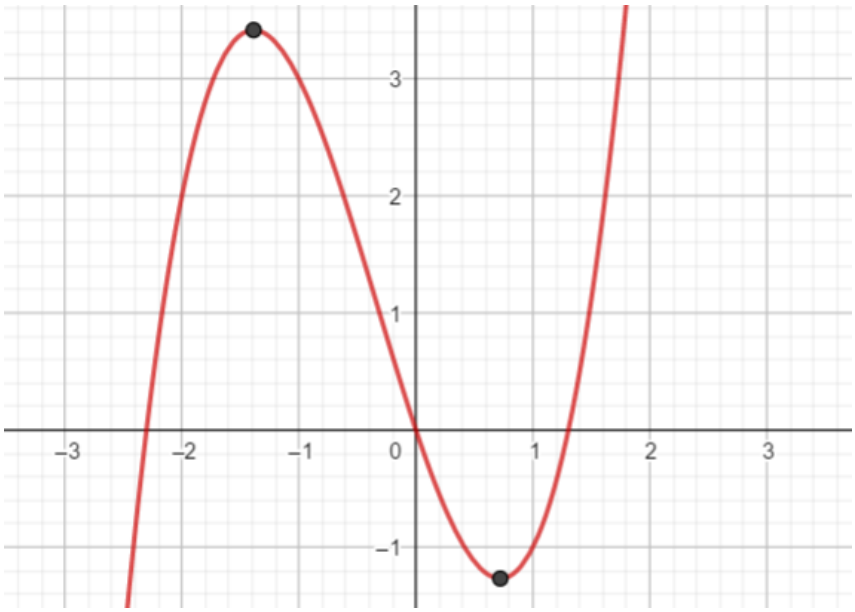
Slika 15: Metoda bisekcije za funkciju $f'(x) = 3x^2 + 2x - 3$ u Octave-u

n	a	b	$f'(a)$	$f'(b)$	x	$f'(x)$	ε
0	0	1	-3	2	0.5	-1.25	0.5
1	0.5	1	-1.25	2	0.75	0.1875	0.25
2	0.5	0.75	-1.25	0.1875	0.625	-0.5781	0.125
3	0.625	0.75	-0.5781	0.1875	0.6875	-0.2070	0.0625
4	0.6875	0.75	-0.2070	0.1875	0.7188	-0.0127	0.0312
5	0.7188	0.75	-0.0127	0.1875	0.7344	0.0867	0.0156
6	0.7188	0.7344	-0.0127	0.0867	0.7266	0.0368	0.0078
7	0.7188	0.7266	-0.0127	0.0368	0.7227	0.0120	0.0039
8	0.7188	0.7227	-0.0127	0.0120	0.7207	-0.00035	0.002
9	0.7207	0.7227	-0.00035	0.0120	0.7217	0.0058	0.0001

Tablica 3: Metoda bisekcije za funkciju $f'(x) = 3x^2 + 2x - 3$

Kao u primjeru 3, u tablici su vrijednosti iteracija zaokružene, a na slici 15 je prikaz rezulata Octave metode za istu funkciju s vrijednostima zaokruženim na 10 decimala.

Funkcija f postiže lokalni minimum u točki $x_{10} = 0.7216796875$ koji iznosi $f(x_{10}) = -1.2683511423$.



Slika 16: Prikaz ekstrema funkcije $f(x) = x^3 + x^2 - 3x$

Iz slike 16 vidi se da se na postavljenom početnom intervalu $[0, 1]$ nalazi jedan od ekstrema zadane funkcije f i da reducirani segment $[0.7207, 0.7227]$ primjenom metode bisekcije odgovara grafu funkcije f. Također, iz slike vidimo da se na analogan način može odrediti i ekstrem na segmentu $[-2, -1]$.

3.1.2. Regula falsi

Regula falsi još se naziva metodom pogrešnog položaja. Ima sigurnu konvergenciju kao i metoda bisekcije, ali u nekim slučajevima je brža od nje. Za razliku od metode bisekcije ne traži se polovište segmenta, već se graf funkcije $y = f(x)$ aproksimira pravcem $y = g(x)$ koji prolazi kroz točke $(a, f(a))$ i $(b, f(b))$, a nultočka tog pravca se označava s x_0. Pravac sigurno siječe os x ako vrijedi $f(a) \cdot f(b) < 0$ i nakon toga točka a ili b pomiče se u x_0, ovisno o tome u kojem intervalu se nalazi nultočka. Na taj način nultočka ostaje unutar novodobivenog segmenta. Postupak se dalje ponavlja na segmentu $[a, x_0]$ ili $[x_0, b]$.
Pošto se traže ekstremi, prilagođavaju se koraci za primjenu ove metode:

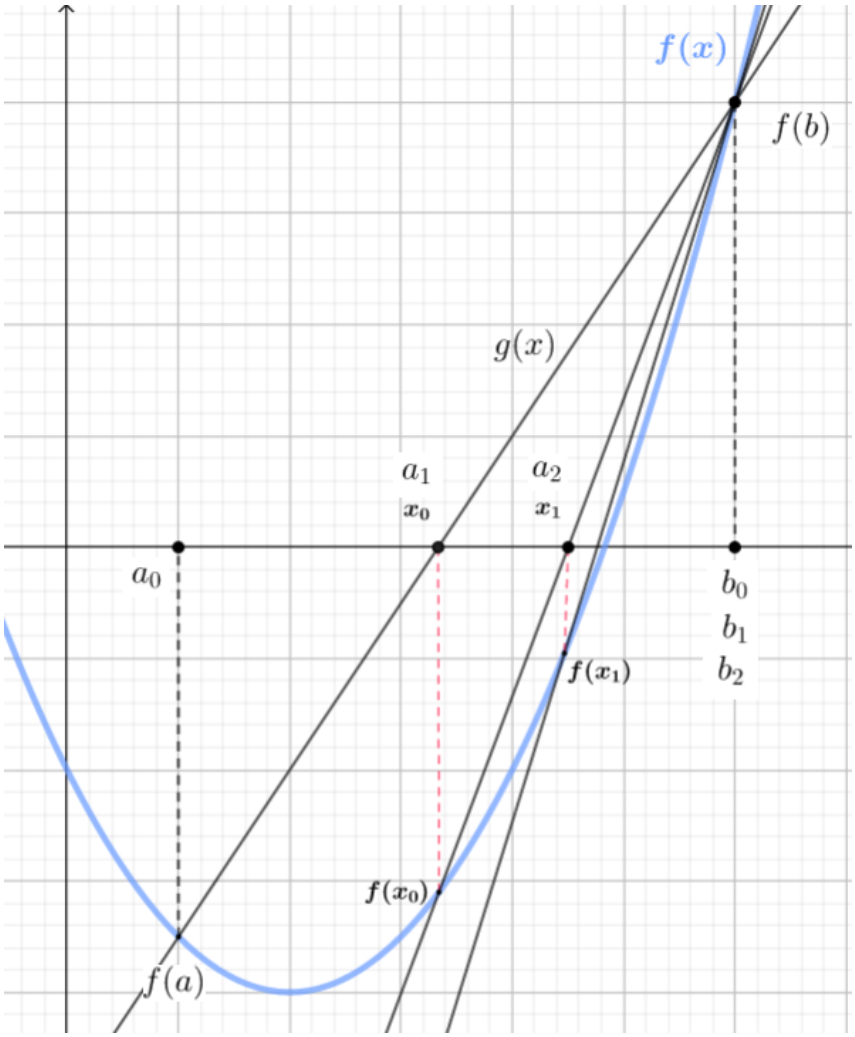
1. Postavlja se početni segment na deriviranu funkciju f' kao što je prikazano u metodi bisekcije.

2. Nultočka pravca računa se pomoću jednadžbe pravca kroz dvije točke iz derivacije funkcije f' i uvjeta $g(x) = 0$.
$$x_0 = b_0 - f'(b_0)\frac{b_0 - a_0}{f'(b_0) - f'(a_0)}$$

3. Pomiče se točka a ili b u x_0, ovisno ako vrijedi
$$f'(a_0) \cdot f'(x_0) < 0 \;\; \Rightarrow \;\; a_1 = a_0, \;\; b_1 = x_0,$$
$$f'(a_0) \cdot f'(x_0) > 0 \;\; \Rightarrow \;\; a_1 = x_0, \;\; b_1 = b_0.$$

4. Iteracijski postupak se ponavlja, sve dok se ne postigne željena točnost ε.
$$x_n = b_n - f'(b_n)\frac{b_n - a_n}{f'(b_n) - f'(a_n)}$$

5. Jedan od općenitih kriterija zaustavljanja koji će se koristiti u ovoj metodi glasi
$$|f'(x_n)| < \varepsilon, \quad |x_n - x_{n-1}| < \delta.$$



Slika 17: Regula falsi

Primjer 5. S točnošću $\varepsilon = 0.001$ korištenjem metode regula falsi treba odrediti ekstreme funkcije $f(x) = x^3 + x^2 - 3x$.

1. Za pronalazak ekstrema potrebna derivacija zadane funkcije f i tada početna funkcija glasi
$$f'(x) = 3x^2 + 2x - 3.$$

2. Kao u metodi bisekcije, rade se koraci za postavljanje segmenta u kojem se nalazi nultočka, odnosno u ovom slučaju ekstrem i iz toga se dobiva segment $[0, 1]$.
$$f'(0) = -3 < 0$$
$$f'(1) = 2 > 0$$

3. Nakon postavljenog segmenta, računa se x_0 pomoću jednadžbe pravca.
Svaka nova iteracija računa se tom jednadžbom koristeći novonastali reducirani segment.
$$x_n = b_n - \frac{b_n - a_n}{f'(b_n) - f'(a_n)} f'(b_n)$$

4. Prije svake iteracije provjerava se ispunjenje uvjeta $|f'(x_n)| < \varepsilon, \ \ |x_n - x_{n-1}| < \delta$.

5. Ako vrijedi uvjet $f'(x_n)f'(b_n) < 0$, tada je $a_{n+1} = x_n$ i $b_{n+1} = b_n$. U protivnom je $a_{n+1} = a_n$ i $b_{n+1} = x_n$.

6. Iteracija se ponavlja sve dok se ne postigne željena točnost.

n	a	b	$f'(a)$	$f'(b)$	x_n	$f'(x_n)$	$\lvert x_n - x_{n-1}\rvert$
0	0	1	-3	2	0.6	-0.72	$-$
1	0.6	1	-0.72	2	0.7059	-0.0934	0.1059
2	0.7059	1	-0.0934	2	0.7190	-0.0111	0.0131
3	0.7190	1	-0.0111	2	0.7206	-0.0013	0.0015
4	0.7206	1	-0.0013	2	0.7207	-0.0002	0.0002

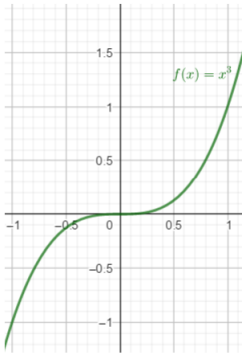
Tablica 4: Prikaz metode regula falsi uz potrebnih n iteracija za funkciju $f'(x) = 3x^2 + 2x - 3$

```
>> [x,itCount]=regula(@funcx1, 0, 1,0.001,0.001)
a=0, fa=-3, x=0.6, fx=-0.72, b=1, fb=2
a=0.6, fa=-0.72, x=0.7058823529, fx=-0.09342560554, b=1, fb=2
a=0.7058823529, fa=-0.09342560554, x=0.7190082645, fx=-0.01106481798, b=1, fb=2
a=0.7190082645, fa=-0.01106481798, x=0.7205542725, fx=-0.001296076036, b=1, fb=2
a=0.7205542725, fa=-0.001296076036, x=0.7207352467, fx=-0.0001516191269, b=1, fb=2

broj iteracija = 5
x =    7.207352466946146e-01
itCount =  5
```

Slika 18: Regula falsi metoda za funkciju $f'(x) = 3x^2 + 2x - 3$ u Octave-u

Vrijednosti iteracija su zaokružene u tablici, a na slici 18 je prikaz rezulata Octave metode za istu funkciju s vrijednostima zaokruženim na 10 decimala. Iz navedenog primjera može se primjetiti da je za istu danu funkciju f, metoda regula falsi brža za 5 iteracija od metode bisekcije što inače ne mora biti slučaj.
Slijedeća slika prikazuje primjer funkcije $f(x) = x^3$ gdje regula falsi nije brža metoda od metode bisekcije.

```
>> [x, itCount] = bisect (@funcx3, -0.2, 0.1, 0.001)
a=-0.2, fa=-0.008, x=-0.05, fx=-0.000125, b=0.1, fb=0.001
a=-0.05, fa=-0.000125, x=0.025, fx=1.5625e-05, b=0.1, fb=0.001
a=-0.05, fa=-0.000125, x=-0.0125, fx=-1.953125e-06, b=0.025, fb=1.5625e-05
a=-0.0125, fa=-1.953125e-06, x=0.00625, fx=2.44140625e-07, b=0.025, fb=1.5625e-05
a=-0.0125, fa=-1.953125e-06, x=-0.003125, fx=-3.051757813e-08, b=0.00625, fb=2.44140625e-07
a=-0.003125, fa=-3.051757813e-08, x=0.0015625, fx=3.814697266e-09, b=0.00625, fb=2.44140625e-
a=-0.003125, fa=-3.051757813e-08, x=-0.00078125, fx=-4.768371582e-10, b=0.0015625, fb=3.81469
a=-0.00078125, fa=-4.768371582e-10, x=0.000390625, fx=5.960464478e-11, b=0.0015625, fb=3.8146
a=-0.00078125, fa=-4.768371582e-10, x=-0.0001953125, fx=-7.450580597e-12, b=0.000390625, fb=5

broj iteracija = 9
x =   -1.953125000000000e-04
itCount =  9
>> [x, itCount] = regula (@funcx3, -0.2, 0.1, 0.001, 0.001)
a=-0.2, fa=-0.008, x=0.06666666667, fx=0.0002962962963, b=0.1, fb=0.001
a=-0.2, fa=-0.008, x=0.05714285714, fx=0.0001865889213, b=0.06666666667, fb=0.0002962962963
a=-0.2, fa=-0.008, x=0.05128205128, fx=0.0001348640402, b=0.05714285714, fb=0.0001865889213
a=-0.2, fa=-0.008, x=0.04711616572, fx=0.0001045947345, b=0.05128205128, fb=0.0001348640402
a=-0.2, fa=-0.008, x=0.04392698099, fx=8.476060901e-05, b=0.04711616572, fb=0.0001045947345
a=-0.2, fa=-0.008, x=0.04136965116, fx=7.080200829e-05, b=0.04392698099, fb=8.476060901e-05
a=-0.2, fa=-0.008, x=0.03925220905, fx=6.047728825e-05, b=0.04136965116, fb=7.080200829e-05
a=-0.2, fa=-0.008, x=0.03745711376, fx=5.255365552e-05, b=0.03925220905, fb=6.047728825e-05
a=-0.2, fa=-0.008, x=0.0359073893, fx=4.629685511e-05, b=0.03745711376, fb=5.255365552e-05
a=-0.2, fa=-0.008, x=0.03455002325, fx=4.124250463e-05, b=0.0359073893, fb=4.629685511e-05
a=-0.2, fa=-0.008, x=0.03334704617, fx=3.70827653e-05, b=0.03455002325, fb=4.124250463e-05
a=-0.2, fa=-0.008, x=0.03227039261, fx=3.360568464e-05, b=0.03334704617, fb=3.70827653e-05
a=-0.2, fa=-0.008, x=0.03129877341, fx=3.06606921e-05, b=0.03227039261, fb=3.360568464e-05

broj iteracija = 13
x =    3.129877340798730e-02
itCount =  13
```

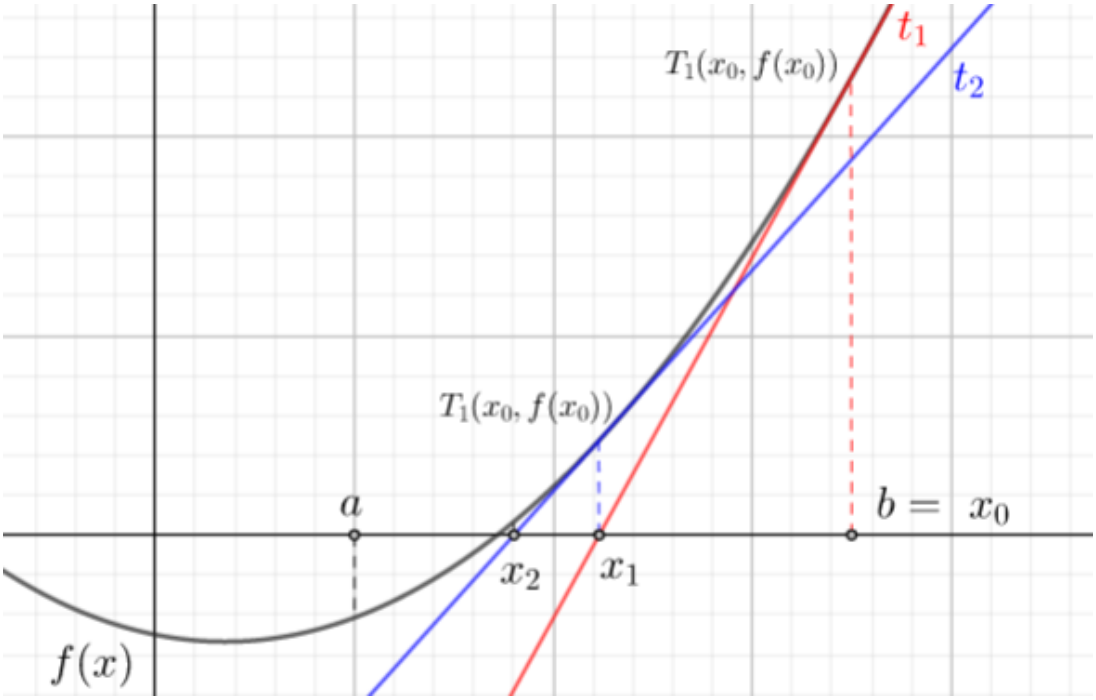


Slika 19: Regula falsi i metoda bisekcije za funkciju $f(x) = x^3$ u Octave-u

Iz slike 19 može se vidjeti da je metoda bisekcije brža za 4 iteracije od regula falsi metode. Egzatna nultočka funkcije je 0, a numerički se može primjetiti da je metoda bisekcije bliža tom rješenju.

3.1.3. Newtonova metoda

Newtonovu metodu još nazivamo metodom tangente jer se graf funkcije f aproksimira tangentom prema nultočki. To se postiže povlačenjem tangente na početnu zadanu točku, te svakom novom aproksimacijom odstupanje od nultočke je sve manje.



Slika 20: Newtonova metoda

Na slici 20 vidi se segment $[a, b]$ i inicijalna početna točka x_0. Povlači se tangenta t_1 u točki $(x_0, f(x_0))$ kako bi se definirala nova aproksimacija x_1 koja siječe os x. Taj postupak se ponavlja za točku x_2, odnosno postupak se ponavlja sve dok se ne postigne željena točnost.
Jednadžba tangente u točki $(x_n, f(x_n))$ glasi

$$y - f(x_n) = f'(x_n)(x - x_n),$$

a kako bi se dobio presjek na osi x u jednadžbu se uvrštava $y = 0$.
Rješavanjem jednadžbe po nepoznanici x dobivamo novu aproksimaciju nultočke

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

Dobivena jednadžba predstavlja svaku novu aproksimaciju za iteracijski postupak Newtonove metode.

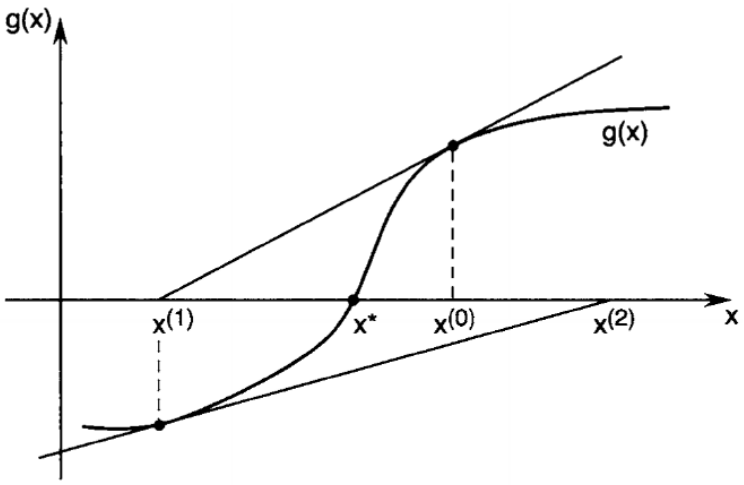
Kao i u prethodnim metodama, jednadžba iteracijskog postupka prilagodit će se traženju ekstrema tako da glasi

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

Pretpostavke kod korištenja Newtonove metode:

- Nultočka treba bit izolirana unutar segmenta $[a, b]$, $f'(x)$ i $f''(x)$ trebaju biti neprekidne i ne smiju mijenjati predznak na tom segmentu.
- Zadana ili izabrana inicijalna točka x_0 treba biti dovoljno blizu rješenja.

Newtonova metoda se ne koristi za pronalazak točne vrijednosti nultočke, već za davanje bolje i brže aproksimacije. Neprikladna je za računanje nultočaka u čijoj okolini funkcija f ima mali nagib jer se javlja problem konvergencije, odnosno nema je. Ako je početna aproksimacija predaleko od nultočke, Newtonova metoda ne mora konvergirati.



Slika 21: Problem konvergencije [3]

Sljedeći teorem govori o globalnoj konvergenciji Newtonove metode.

Teorem 8. *Neka je $f \in C^2[a, b]$, $f(a)f(b) < 0$ i neka f' i f'' nemaju nultočke u $[a, b]$, (tj. f' i f'' imaju fiksni predznak na $[a, b]$). Ako polazna iteracija x_0 iz segmenta $[a, b]$ zadovoljava uvjet*

$$f(x_0)f''(x_0) > 0,$$

onda niz iteracija dobiven Newtonovom metodom konvergira prema (jedinstvenoj jednostrukoj) nultočki α funkcije f.

Primjer 6. S točnošću $\varepsilon = 0.001$ korištenjem Newtonove metode treba odrediti ekstreme funkcije $f(x) = x^3 + x^2 - 3x$ (kao i u prethodnim metodama).

1. Za pronalazak ekstrema potrebna je derivacija zadane funkcije f i tada početna funkcija glasi
$$f'(x) = 3x^2 + 2x - 3.$$
2. Kao i u predhodnim metodama, rade se koraci za postavljanje segmenta u kojem se

nalaze nultočke, odnosno u ovom slučaju ekstremi i iz toga se dobiva segment $[0, 1]$.

$$f'(0) = -3 < 0$$
$$f'(1) = 2 > 0$$

Suprotne vrijednosti javljaju se na intervalu $[0, 1]$, nultočka je pozitivna i vrijedi uvjet $f(a)f(b) < 0$.

3. Odabire se inicijalna točka iz segmenta $x_0 = 1$ i izvodi se derivacija početne funkcije. Pošto je početna funkcija već derivacija, zbog traženja ekstrema, potrebna je druga derivacija funkcije
$$f''(x) = 6x + 2$$
u koju se uvrštava $x_0 = 1$.

4. Računa se sljedeća iteracija
$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)},$$
sve dok se ne ispuni uvjet $|f'(x_n)| < \varepsilon, \ \ |x_n - x_{n-1}| < \delta$.

n	x_n	$f'(x_n)$	$f''(x_n)$	$\lvert x_n - x_{n-1}\rvert$
0	1	2	8	–
1	0.75	0.1875	6.5	0.25
2	0.721154	0.002496	6.326923	0.028846
3	0.720759	0.0000005	6.324556	0.003946

Tablica 5: Prikaz Newtonove metode sa potrebnih n iteracija za funkciju $f'(x) = 3x^2 + 2x - 3$

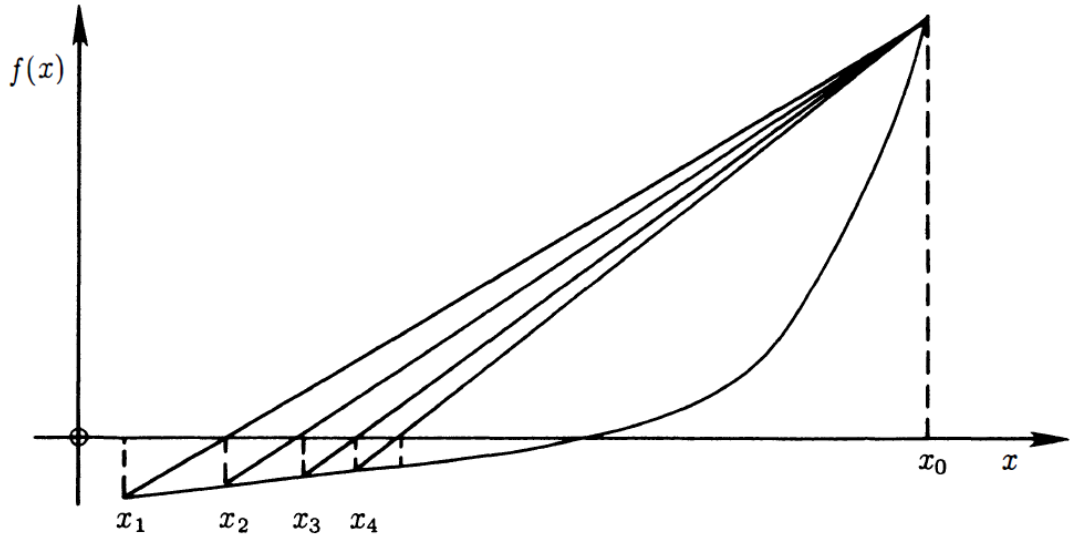
```
>> [x,itCount]=newton(@funcx1, @funcx2,1,0.001,0.001)
x=1, f(x)=2, f'(x)=8, error estimate=-
x=0.75, f(x)=0.1875, f'(x)=6.5, error estimate=0.25
x=0.7211538462, f(x)=0.002496301775, f'(x)=6.326923077, error estimate=0.02884615385
x=0.7207592939, f(x)=4.670144484e-07, f'(x)=6.324555763, error estimate=0.0003945522563

broj iteracija = 4
x =    7.207592200561290e-01
itCount =  4
```

Slika 22: Newtonova metoda za funkciju $f(x) = 3x^2 + 2x - 3$ u Octave-u

Vrijednosti iteracija su zaokružene u tablici, a na slici 22 je prikaz rezultata iste funkcije u Octave metodi, gdje su vrijednosti zaokružene na 10 decimala.
Može se primjetiti da je potrebno 4 iteracije za istu zadanu funkciju i da je na ovom primjeru Newtonova metoda bila brža za 1 iteraciju od regule falsi metode.

3.1.4. Prednosti i nedostaci indirektnih metoda

Regula falsi i metoda bisekcije dijele sličnosti jer su zatvorene metode čijom primjenom se dobiva reducirani segment unutar kojeg se nalazi nultočka, odnosno ekstrem. Newtonova metoda daje približno rješenje nultočke koje se dobiva iteracijskim postupkom počevši od jedne inicijalne točke, dok ove druge zahtijevaju inicijalni segment koji se postupno smanjuje.
Metoda bisekcije ima sigurnu, ali sporu konvergenciju, dok regula falsi uvijek linearno konvergira, ali u nekim slučajevima može biti i jako spora. Glavni nedostatak regule falsi jest što ne garantira bržu konvergenciju, te ovisi o početnom segmentu.

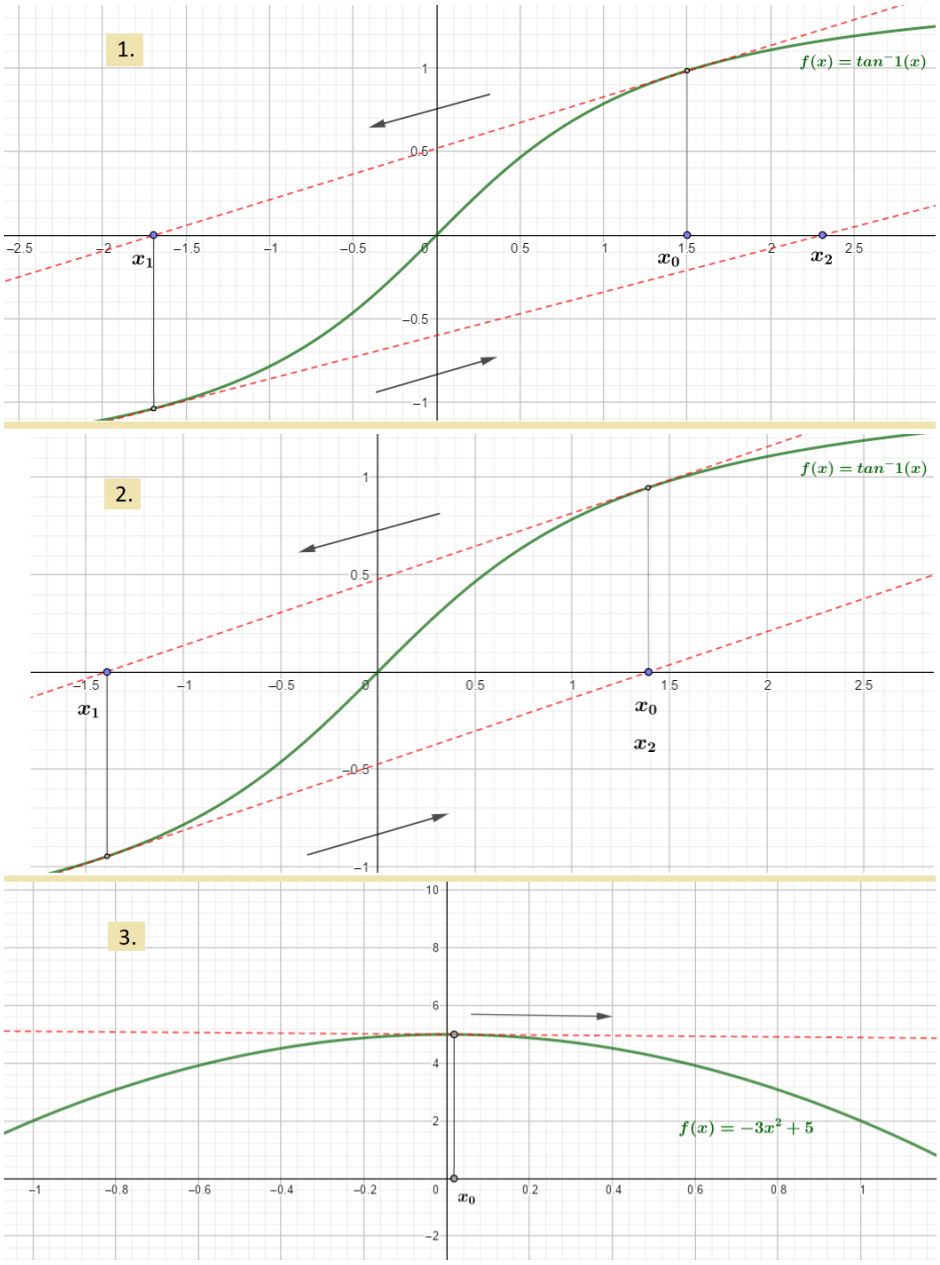


Slika 23: Problem spore konvergencije metode regula falsi [3]

Newtonova metoda jedna je od najbržih metoda koja ima brzu, ali nepouzdanu konvergenciju, dok metoda bisekcije ima sporu, ali sigurnu konvergenciju. Kod Newtonove metode što se više približava nultočki, broj znamenki se značajno aproksimira svakim korakom, odnosno konvergencija je kvadrična, a kod bisekcije konvergencija se smanjuje za pola segmenta. Za razliku od bisekcije, Newtonova metoda zahtijeva derivaciju zadane funkcije, dok to nije potrebno kod metoda bisekcije, niti regule falsi. U slučaju traženja ekstrema kod obje metode je potrebna derivacija funkcije, a kod Newtonove metode čak i druga derivacija. Newtonova metoda se može koristiti i u višedimenzionalnoj optimizaciji, te se u većini slučajeva najčešće i koristi.

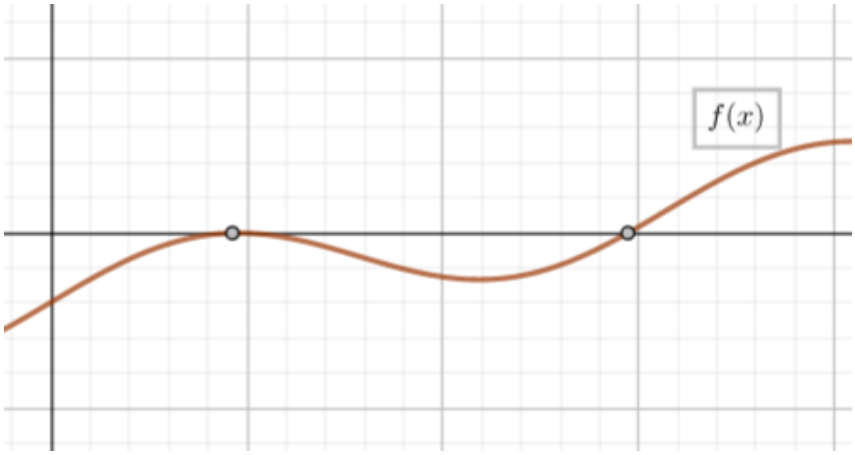
Problemi kod Newtonove metode ovise i o početnoj aproksimaciji nultočke:

1. Kada je početna aproksimacija predaleko od lokalno tražene nultočke, Newtonova metoda može divergirati.
2. Pogrešno odabrana početna aproksimacija može uzrokovati cikliranje.
3. Kada je početna aproksimacija previše blizu rješenja jednadžbe $f'(x) = 0$ (odnosno $f''(x) = 0$ u slučaju traženja ekstrema), tada može nastupiti problem s aritmetikom realnih brojeva na računalu.



Slika 24: Problemi kod Newtonove metode [4]

Na slici 24 u prva dva grafa prikazana je funkcija $f(x) = \operatorname{arctg} x$ s različitim startnim točkama $x_0 = 1.5$ i $x_0 = 1.39$, a treći graf prikazuje funkciju $f(x) = -3x^2 + 5$ sa startnom točkom $x_0 = 0.02$. Oni predstavljaju moguće probleme Newtonove metode koji se mogu dogoditi kod pogrešnog odabira početne aproksimacije.
Rješenje nekih od ovih problema je kombiniranje rješenja s metodom bisekcije za odabir nove početne aproksimacije. Ako se primijeti da iteracija ne ostaje unutar postavljenog segmenta, tada se dobivena vrijednost zanemaruje i zamjenjuje se postupak s metodom bisekcije.



Slika 25: Problem metode bisekcije

Na slici 25 vide se dvije nultočke funkcije f od kojih se metodom bisekcije može dobiti samo jedna (desna) jer zahtijeva da u intervalu u kojem je nultočka, funkcija mijenja predznak.
Svaka od ovih metoda može se kombinirati s obzirom na gore navedene probleme, kao što je npr. pogrešno odabrana početna aproksimacija ili pogrešno odabran početni segment, itd.

3.1.5. Implementacija indirektnih metoda u Octave programskom jeziku

```
1  ## Created: 2020-06-01
2  ## source: http://www.math.pitt.edu/~sussmanm/2070/lab_03/index.html
3
4  function [x, itCount] = bisect ( func, a, b, epsilon, iter=100)
5  fa = func(a);
6  fb = func(b);
7
8  if (fa * fb > 0)
9    error('Interval nije dobro odabran!');
10 endif
11
12 for itCount = 1:iter
13   x = (b + a) / 2;
14   fx = func(x);
15   % ispis trenutnog koraka iteracije
16   disp(strcat(  'a=' , num2str(a,10), ', fa=' , num2str(fa,10), ...
17               ', x=' , num2str(x,10), ', fx=' , num2str(fx,10), ...
18               ', b=' , num2str(b,10), ', fb=' , num2str(fb,10)))
19
20   if ( fx == 0 ) || (abs (b - x) < epsilon)
21     printf('\nbroj iteracija = %i\n', itCount);
22     return;
23   endif
24
25   if ( sign(fa) * sign(fx) <= 0 )
26     b = x;
27     fb = fx;
28   else
29     a = x;
30     fa = fx;
31   endif
32 endfor
33 error(sprintf('bisekcija nije uspjela sa %i iteracija.', iter));
```

Algoritam 3.1: Octave kod za metodu bisekcije [5]

Algoritam 3.1 prikazuje metodu bisekcije koja prima parametar za funkciju $f(x)$, parametre a i b za promatrani segment, te parametar ε kojim se zadaje željena točnost.
Također na liniji 25 u kodu metode 3.1 može se primijetiti kriterij zaustavljanja koji je specifičan samo za metodu bisekcije, ali prikazat će se i drugi kriterij koji je naveden u sekciji 3.1. pod nazivom dinamička ocjena greške.

```
1  ## Created: 2020-09-12
2  function [x, itCount] = bisect1 ( func, func1, a, b, epsilon, iter=100)
3  fa = func(a);
4  fb = func(b);
5  M = minDer(func1,a,b, 0.001);
6
7  if (fa * fb > 0)
8    error('Interval nije dobro odabran!');
9  endif
10
11 for itCount = 1:iter
12   x = (b + a) / 2;
13   fx = func(x);
14   % ispis trenutnog koraka iteracije
15   disp(strcat('a=' , num2str(a,10), ', fa=' , num2str(fa,10), ...
16               ', x=' , num2str(x,10), ', fx=' , num2str(fx,10), ...
17               ', b=' , num2str(b,10), ', fb=' , num2str(fb,10)))
18
19   if ( fx == 0 ) || ( abs (fx) < M * epsilon )
20     printf('\nbroj iteracija = %i\n', itCount);
21     return;
22   endif
23
24   if ( sign(fa) * sign(fx) <= 0 )
25     b = x;
26     fb = fx;
27   else
28     a = x;
29     fa = fx;
30   endif
31 endfor
32 error(sprintf('bisekcija nije uspjela sa %i iteracija.', iter));
```

Algoritam 3.2: Octave kod za metodu bisekcije s kriterijem dinamičke ocjene

Algoritam 3.2 na liniji 19 prikazuje kriterij zaustavljanja dinamičke ocjene za koji vrijedi

$$|f(x_n)| < m_1\varepsilon, \quad m_1 = \min_{x\in[a,b]} |f'(x)|,$$

a m_1 se računa u funkciji `minDer` koja se poziva na liniji 4.

```
1  function rez = minDer(f, a, b, delta)
2    rez = abs(f(a));
3    xi = a + delta;
4    while (xi < b)
5      temp = abs(f(xi));
6      if (rez > temp)
7        rez = temp;
8      endif
9      xi += delta;
10   endwhile
11 endfunction
```

Algoritam 3.3: Octave kod za izračun m_1

Primjer 7. Potražimo nultočke metodom bisekcije za funkciju $f(x) = x^3 - 1.5$ s točnošću manjom od $\varepsilon = 10^{-8}$ na intervalu $[1, 2]$ ([4]).

```
>> [x,itCount]=bisect(@func, 1, 2 ,1.0e-8)
a=1, fa=-0.5, x=1.5, fx=1.875, b=2, fb=6.5
a=1, fa=-0.5, x=1.25, fx=0.453125, b=1.5, fb=1.875
a=1, fa=-0.5, x=1.125, fx=-0.076171875, b=1.25, fb=0.453125
a=1.125, fa=-0.076171875, x=1.1875, fx=0.1745605469, b=1.25, fb=0.453125
a=1.125, fa=-0.076171875, x=1.15625, fx=0.04580688477, b=1.1875, fb=0.1745605469
a=1.125, fa=-0.076171875, x=1.140625, fx=-0.01601791382, b=1.15625, fb=0.04580688477
a=1.140625, fa=-0.01601791382, x=1.1484375, fx=0.01468420029, b=1.15625, fb=0.04580688477
a=1.140625, fa=-0.01601791382, x=1.14453125, fx=-0.0007192492485, b=1.1484375, fb=0.014684200
a=1.14453125, fa=-0.0007192492485, x=1.146484375, fx=0.006969355047, b=1.1484375, fb=0.014684
a=1.14453125, fa=-0.0007192492485, x=1.145507812, fx=0.003121775575, b=1.146484375, fb=0.0069
a=1.14453125, fa=-0.0007192492485, x=1.145019531, fx=0.001200444181, b=1.145507812, fb=0.0031
a=1.14453125, fa=-0.0007192492485, x=1.144775391, fx=0.0002403927647, b=1.145019531, fb=0.001
a=1.14453125, fa=-0.0007192492485, x=1.14465332, fx=-0.0002394794119, b=1.144775391, fb=0.000
a=1.14465332, fa=-0.0002394794119, x=1.144714355, fx=4.438832093e-07, b=1.144775391, fb=0.000
a=1.14465332, fa=-0.0002394794119, x=1.144683838, fx=-0.0001195209626, b=1.144714355, fb=4.43
a=1.144683838, fa=-0.0001195209626, x=1.144699097, fx=-5.953933924e-05, b=1.144714355, fb=4.4
a=1.144699097, fa=-5.953933924e-05, x=1.144706726, fx=-2.954792791e-05, b=1.144714355, fb=4.4
a=1.144706726, fa=-2.954792791e-05, x=1.144710541, fx=-1.455207232e-05, b=1.144714355, fb=4.4
a=1.144710541, fa=-1.455207232e-05, x=1.144712448, fx=-7.054107049e-06, b=1.144714355, fb=4.4
a=1.144712448, fa=-7.054107049e-06, x=1.144713402, fx=-3.305115043e-06, b=1.144714355, fb=4.4
a=1.144713402, fa=-3.305115043e-06, x=1.144713879, fx=-1.430616698e-06, b=1.144714355, fb=4.4
a=1.144713879, fa=-1.430616698e-06, x=1.144714117, fx=-4.933669395e-07, b=1.144714355, fb=4.4
a=1.144714117, fa=-4.933669395e-07, x=1.144714236, fx=-2.474191385e-08, b=1.144714355, fb=4.4
a=1.144714236, fa=-2.474191385e-08, x=1.144714296, fx=2.095706355e-07, b=1.144714355, fb=4.43
a=1.144714236, fa=-2.474191385e-08, x=1.144714266, fx=9.241435772e-08, b=1.144714296, fb=2.09
a=1.144714236, fa=-2.474191385e-08, x=1.144714251, fx=3.383622116e-08, b=1.144714266, fb=9.24
a=1.144714236, fa=-2.474191385e-08, x=1.144714244, fx=4.547153321e-09, b=1.144714251, fb=3.38

broj iteracija = 27
x =  1.144714243710041
itCount =  27
```

Slika 26: Pozivanje funkcije `bisect` i izlazne vrijednosti

Slika 26 prikazuje poziv funkcije `bisect` i izlazne vrijednosti koje vraća funkcija. Iz toga, može se zaključiti da je bilo potrebno 27 iteracija za $x_{27} \approx 1.1447$.
Uočimo u pozivu funkcije `bisect` parametar `@func` koji je također pozivajuća funkcija, tj. pointer na funkciju.

```
1 ## Created: 2020-06-01
2
3 function [f]=func(x)
4
5 f = x.^3 - 1.5;
6
7 endfunction
```

Algoritam 3.4: Pozivajuća funkcija iz primjera 7

Za funkciju `bisect` nije bila potrebna derivacija funkcije f, ali zato će se koristiti za funkciju `bisect1` pošto je za kriterij zaustavljanja preko dinamičke ocjene potrebna derivacija.

```
1 ## Created: 2020-06-01
2
3 function [f]=func1(x)
4
5 f = 3*x.^2;
6
7 endfunction
```

Algoritam 3.5: Pozivajuća derivacija funkcije iz primjera 7

```
>> [x,itCount]=bisect1(@func, @funcl, 1, 2 ,1.0e-8)
a=1, fa=-0.5, x=1.5, fx=1.875, b=2, fb=6.5
a=1, fa=-0.5, x=1.25, fx=0.453125, b=1.5, fb=1.875
a=1, fa=-0.5, x=1.125, fx=-0.076171875, b=1.25, fb=0.453125
a=1.125, fa=-0.076171875, x=1.1875, fx=0.1745605469, b=1.25, fb=0.453125
a=1.125, fa=-0.076171875, x=1.15625, fx=0.04580688477, b=1.1875, fb=0.1745605469
a=1.125, fa=-0.076171875, x=1.140625, fx=-0.01601791382, b=1.15625, fb=0.04580688477
a=1.140625, fa=-0.01601791382, x=1.1484375, fx=0.01468420029, b=1.15625, fb=0.04580688477
a=1.140625, fa=-0.01601791382, x=1.14453125, fx=-0.0007192492485, b=1.1484375, fb=0.014684200
a=1.14453125, fa=-0.0007192492485, x=1.146484375, fx=0.006969355047, b=1.1484375, fb=0.014684
a=1.14453125, fa=-0.0007192492485, x=1.145507812, fx=0.003121775575, b=1.146484375, fb=0.0069
a=1.14453125, fa=-0.0007192492485, x=1.145019531, fx=0.001200444181, b=1.145507812, fb=0.0031
a=1.14453125, fa=-0.0007192492485, x=1.144775391, fx=0.0002403927647, b=1.145019531, fb=0.001
a=1.14453125, fa=-0.0007192492485, x=1.14465332, fx=-0.0002394794119, b=1.144775391, fb=0.000
a=1.14465332, fa=-0.0002394794119, x=1.144714355, fx=4.438832093e-07, b=1.144775391, fb=0.000
a=1.14465332, fa=-0.0002394794119, x=1.144683838, fx=-0.0001195209626, b=1.144714355, fb=4.43
a=1.144683838, fa=-0.0001195209626, x=1.144699097, fx=-5.953933924e-05, b=1.144714355, fb=4.4
a=1.144699097, fa=-5.953933924e-05, x=1.144706726, fx=-2.954792791e-05, b=1.144714355, fb=4.4
a=1.144706726, fa=-2.954792791e-05, x=1.144710541, fx=-1.455207232e-05, b=1.144714355, fb=4.4
a=1.144710541, fa=-1.455207232e-05, x=1.144712448, fx=-7.054107049e-06, b=1.144714355, fb=4.4
a=1.144712448, fa=-7.054107049e-06, x=1.144713402, fx=-3.305115043e-06, b=1.144714355, fb=4.4
a=1.144713402, fa=-3.305115043e-06, x=1.144713879, fx=-1.430616698e-06, b=1.144714355, fb=4.4
a=1.144713879, fa=-1.430616698e-06, x=1.144714117, fx=-4.933669395e-07, b=1.144714355, fb=4.4
a=1.144714117, fa=-4.933669395e-07, x=1.144714236, fx=-2.474191385e-08, b=1.144714355, fb=4.4

broj iteracija = 23
x =  1.144714236259460
itCount =  23
```

Slika 27: Pozivanje funkcije `bisect1` i izlazne vrijednosti

Vidimo da promjena kriterija zaustavljanja utječe na broj potrebnih iteracija primjenom iste metode. S kriterijem dinamičke ocjene koju koristi funkcija `bisect1` potrebne su 23 iteracije za $x_{23} \approx 1.1447$.

```
## Created: 2020-09-12
## source: http://www.math.pitt.edu/~sussmanm/2070/lab_03/index.html

function [x, itCount] = regula (func, a, b, delta, epsilon, iter=100)
fa = func(a);
fb = func(b);
xs = a;
if (fa * fb > 0)
  error('Interval nije dobro odabran!');
endif

for itCount = 1:iter
  x = (a * fb - b * fa) / (fb - fa);
  fx = func(x);

  disp(strcat(  'a=' , num2str(a,10), ', fa=' , num2str(fa,10),
              ', x=' , num2str(x,10), ', fx=' , num2str(fx,10),
              ', b=' , num2str(b,10), ', fb=' , num2str(fb,10)))

  if ( fx == 0 ) || ((abs(fx) < epsilon) &&  (abs(x - xs) < delta))
       printf('\nbroj iteracija = %i\n', itCount);
    return;
  endif

  if ( sign(fa) * sign(fx) <= 0 )
    b = x;
    fb = fx;
```

```
28    else
29      a = x;
30      fa = fx;
31    endif
32    xs = x;
33  endfor
34  error(sprintf('metoda regula falsi nije uspjela sa %i iteracija.', iter));
```

Algoritam 3.6: Octave kod za metodu regula falsi

Algoritam 3.6 prikazuje funkciju `regula` koja koristi metodu regula falsi s parametrima istim kao i kod metode bisekcije, a jedina razlika u kodu je kod računanja nultočke pravca na liniji 13. Također, na liniji 20 u kodu metode 3.6 je kriterij zaustavljanja koji se može koristiti u svim metodama i prikazat će se primjena kriterija dinamičke ocjene na funkciji `regula1`, kao što je prikazan kod metode bisekcije.

```
1  ## Created: 2020-09-12
2  ## source: http://www.math.pitt.edu/~sussmanm/2070/lab_03/index.html
3
4  function [x, itCount] = regula1 ( func, func1, a, b, epsilon, iter=100)
5  fa = func(a);
6  fb = func(b);
7  M = minDer(func1, a,b, 0.001);
8  if (fa * fb > 0)
9    error('Interval nije dobro odabran!');
10 endif
11
12 for itCount = 1:iter
13   x = (a * fb - b * fa) / (fb - fa);
14   fx = func(x);
15
16   disp(strcat(  'a=' , num2str(a,10), ', fa=' , num2str(fa,10), ...
17               ', x=' , num2str(x,10), ', fx=' , num2str(fx,10), ...
18               ', b=' , num2str(b,10), ', fb=' , num2str(fb,10)))
19   if ( fx == 0 ) ||  (abs(fx) < M * epsilon)
20     printf('\nbroj iteracija = %i\n', itCount);
21     return;
22   endif
23
24   if ( sign(fa) * sign(fx) <= 0 )
25     b = x;
26     fb = fx;
27   else
28     a = x;
29     fa = fx;
30   endif
31 endfor
32 error(sprintf('metoda regula falsi nije uspjela sa %i iteracija.', iter));
```

Algoritam 3.7: Octave kod za metodu regula falsi s primjenom kriterija dinamičke ocjene

```
>> [x,itCount]=regula(@func, 1, 2, 1.0e-8, 1.0e-8)
a=1, fa=-0.5, x=1.071428571, fx=-0.2700437318, b=2, fb=6.5
a=1.071428571, fa=-0.2700437318, x=1.10846746, fx=-0.138025911, b=2, fb=6.5
a=1.10846746, fa=-0.138025911, x=1.127005289, fx=-0.06854446364, b=2, fb=6.5
a=1.127005289, fa=-0.06854446364, x=1.136115215, fx=-0.033550445, b=2, fb=6.5
a=1.136115215, fa=-0.033550445, x=1.140551351, fx=-0.01630535227, b=2, fb=6.5
a=1.140551351, fa=-0.01630535227, x=1.142701897, fx=-0.00789685762, b=2, fb=6.5
a=1.142701897, fa=-0.00789685762, x=1.143742166, fx=-0.003818099378, b=2, fb=6.5
a=1.143742166, fa=-0.003818099378, x=1.144244836, fx=-0.001844532826, b=2, fb=6.5
a=1.144244836, fa=-0.001844532826, x=1.144487609, fx=-0.0008907474629, b=2, fb=6.5
a=1.144487609, fa=-0.0008907474629, x=1.14460483, fx=-0.0004300710713, b=2, fb=6.5
a=1.14460483, fa=-0.0004300710713, x=1.144661424, fx=-0.0002076280374, b=2, fb=6.5
a=1.144661424, fa=-0.0002076280374, x=1.144688745, fx=-0.000100233413, b=2, fb=6.5
a=1.144688745, fa=-0.000100233413, x=1.144701934, fx=-4.838711691e-05, b=2, fb=6.5
a=1.144701934, fa=-4.838711691e-05, x=1.144708301, fx=-2.335836752e-05, b=2, fb=6.5
a=1.144708301, fa=-2.335836752e-05, x=1.144711374, fx=-1.127594797e-05, b=2, fb=6.5
a=1.144711374, fa=-1.127594797e-05, x=1.144712858, fx=-5.443304053e-06, b=2, fb=6.5
a=1.144712858, fa=-5.443304053e-06, x=1.144713574, fx=-2.627674822e-06, b=2, fb=6.5
a=1.144713574, fa=-2.627674822e-06, x=1.14471392, fx=-1.268470589e-06, b=2, fb=6.5
a=1.14471392, fa=-1.268470589e-06, x=1.144714087, fx=-6.123349756e-07, b=2, fb=6.5
a=1.144714087, fa=-6.123349756e-07, x=1.144714167, fx=-2.955954013e-07, b=2, fb=6.5
a=1.144714167, fa=-2.955954013e-07, x=1.144714206, fx=-1.426941791e-07, b=2, fb=6.5
a=1.144714206, fa=-1.426941791e-07, x=1.144714225, fx=-6.888344006e-08, b=2, fb=6.5
a=1.144714225, fa=-6.888344006e-08, x=1.144714234, fx=-3.325243103e-08, b=2, fb=6.5
a=1.144714234, fa=-3.325243103e-08, x=1.144714238, fx=-1.605210342e-08, b=2, fb=6.5
a=1.144714238, fa=-1.605210342e-08, x=1.144714241, fx=-7.74890796e-09, b=2, fb=6.5

broj iteracija = 25
x =  1.144714240582157
itCount =  25
```

Slika 28: Pozivanje funkcije `regula` i izlazne vrijednosti

```
>> [x,itCount]=regula1(@func, @func1, 1, 2, 1.0e-8)
a=1, fa=-0.5, x=1.071428571, fx=-0.2700437318, b=2, fb=6.5
a=1.071428571, fa=-0.2700437318, x=1.10846746, fx=-0.138025911, b=2, fb=6.5
a=1.10846746, fa=-0.138025911, x=1.127005289, fx=-0.06854446364, b=2, fb=6.5
a=1.127005289, fa=-0.06854446364, x=1.136115215, fx=-0.033550445, b=2, fb=6.5
a=1.136115215, fa=-0.033550445, x=1.140551351, fx=-0.01630535227, b=2, fb=6.5
a=1.140551351, fa=-0.01630535227, x=1.142701897, fx=-0.00789685762, b=2, fb=6.5
a=1.142701897, fa=-0.00789685762, x=1.143742166, fx=-0.003818099378, b=2, fb=6.5
a=1.143742166, fa=-0.003818099378, x=1.144244836, fx=-0.001844532826, b=2, fb=6.5
a=1.144244836, fa=-0.001844532826, x=1.144487609, fx=-0.0008907474629, b=2, fb=6.5
a=1.144487609, fa=-0.0008907474629, x=1.14460483, fx=-0.0004300710713, b=2, fb=6.5
a=1.14460483, fa=-0.0004300710713, x=1.144661424, fx=-0.0002076280374, b=2, fb=6.5
a=1.144661424, fa=-0.0002076280374, x=1.144688745, fx=-0.000100233413, b=2, fb=6.5
a=1.144688745, fa=-0.000100233413, x=1.144701934, fx=-4.838711691e-05, b=2, fb=6.5
a=1.144701934, fa=-4.838711691e-05, x=1.144708301, fx=-2.335836752e-05, b=2, fb=6.5
a=1.144708301, fa=-2.335836752e-05, x=1.144711374, fx=-1.127594797e-05, b=2, fb=6.5
a=1.144711374, fa=-1.127594797e-05, x=1.144712858, fx=-5.443304053e-06, b=2, fb=6.5
a=1.144712858, fa=-5.443304053e-06, x=1.144713574, fx=-2.627674822e-06, b=2, fb=6.5
a=1.144713574, fa=-2.627674822e-06, x=1.14471392, fx=-1.268470589e-06, b=2, fb=6.5
a=1.14471392, fa=-1.268470589e-06, x=1.144714087, fx=-6.123349756e-07, b=2, fb=6.5
a=1.144714087, fa=-6.123349756e-07, x=1.144714167, fx=-2.955954013e-07, b=2, fb=6.5
a=1.144714167, fa=-2.955954013e-07, x=1.144714206, fx=-1.426941791e-07, b=2, fb=6.5
a=1.144714206, fa=-1.426941791e-07, x=1.144714225, fx=-6.888344006e-08, b=2, fb=6.5
a=1.144714225, fa=-6.888344006e-08, x=1.144714234, fx=-3.325243103e-08, b=2, fb=6.5
a=1.144714234, fa=-3.325243103e-08, x=1.144714238, fx=-1.605210342e-08, b=2, fb=6.5

broj iteracija = 24
x =  1.144714238469983
itCount =  24
```

Slika 29: Pozivanje funkcije `regula1` i izlazne vrijednosti

Korištenjem različitih kriterija zaustavljanja, može se primjetiti da se broj iteracija nije značajno smanjio kod metode regula falsi.

```
## Created: 2020-06-01
## source: http://www.math.pitt.edu/~sussmanm/2070/lab_04/index.html

function [x,itCount]=newton(func, func1, x, delta, epsilon, iter=100)
errorEstimate = '-';

for itCount = 1:iter
  f = func(x);
  fprime = func1(x);
  increment = -f/fprime;

  disp(strcat('x=', num2str(x,10),
              ', f(x)=' , num2str(f,10),
              ', f''(x)=' , num2str(fprime,10),
              ', error estimate=', num2str(errorEstimate,10)
              ));
  errorEstimate = abs(( x + increment ) - x ) ; % errorEstimate |x(n)-x(n-1)|
  x = x + increment;

  if (errorEstimate < delta) && (abs(f) < epsilon)
    printf('\nbroj iteracija = %i\n', itCount);
    return;
  endif

endfor
error('newtonova metoda nije uspjela sa %i iteracija.', iter)
```

Algoritam 3.8: Octave kod za Newton metodu [6]

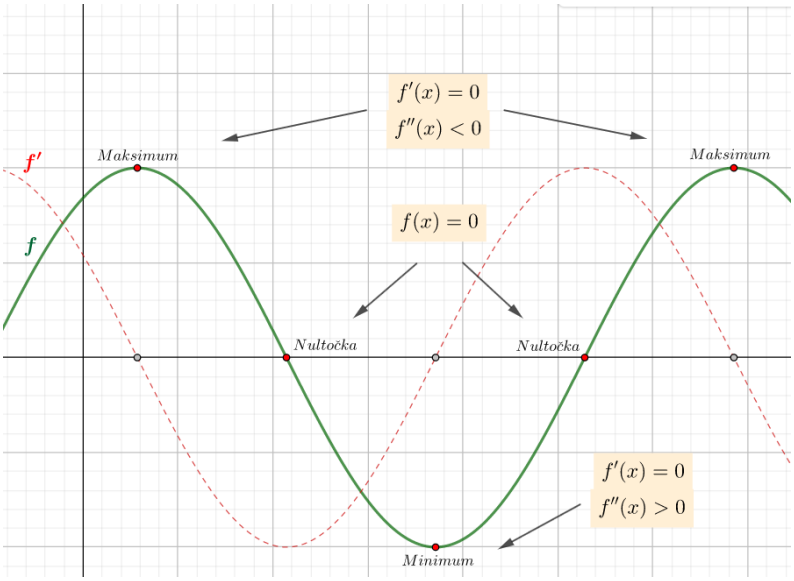
```
>> [x,itCount]=newton(@func, @func1,1,1.0e-8,1.0e-8)
x=1, f(x)=-0.5, f'(x)=3, error estimate=-
x=1.166666667, f(x)=0.08796296296, f'(x)=4.083333333, error estimate=0.1666666667
x=1.145124717, f(x)=0.001614197989, f'(x)=3.933931849, error estimate=0.02154195011
x=1.14471439, f(x)=5.783385035e-07, f'(x)=3.931113102, error estimate=0.0004103268817
x=1.144714243, f(x)=7.416289804e-14, f'(x)=3.931112091, error estimate=1.47118256e-07

broj iteracija = 5
x =  1.144714242553332
itCount =  5
```

Slika 30: Pozivanje funkcije `newton` i izlazne vrijednosti

Vidi se da Newtonova metoda zahtijeva puno manje iteracija, svega 5, za vrijednost $x_5 \approx 1.1447$.

3.2. Direktne metode pretraživanja

Glavna razlika metoda pretraživanja je da se kod direktnih metoda traže izolirani ekstremi, minimum ili maksimum funkcije, dok se kod indirektnih metoda traži nultočka funkcije pomoću koje se traži lokalni ekstrem.



Slika 31: Uvjeti traženja nultočke i ekstrema funkcije

Za traženje lokanih ekstrema trebaju biti ispunjeni uvjeti:

- $f'(x) = 0$
- $f''(x) < 0$ za lokalni maksimum ili $f''(x) > 0$ za lokalni minimum

Postoje slučajevi u kojima se ne može analitički doći do rješenja $f'(x) = 0$ i zbog toga se koriste direktne metode. Prvo se procjenjuje segment u kojem se nalazi lokalni ekstrem kao što se radilo u indirektnim metodam ili se segment procjenjuje grafički. Kada je poznat segment u kojem se nalazi ekstrem, iterativnim postupkom dobiva se željena aproksimacija ekstrema.

Direktne metode pretraživanja koriste se u sljedećim slučajevima:

- kada funkcija f nije derivabilna,
- kada je f' komplicirana za izračun ili ne postoji,
- kada je lokacija optimalnog rješenja poznata u grubo.

3.2.1. Metoda zlatnog reza

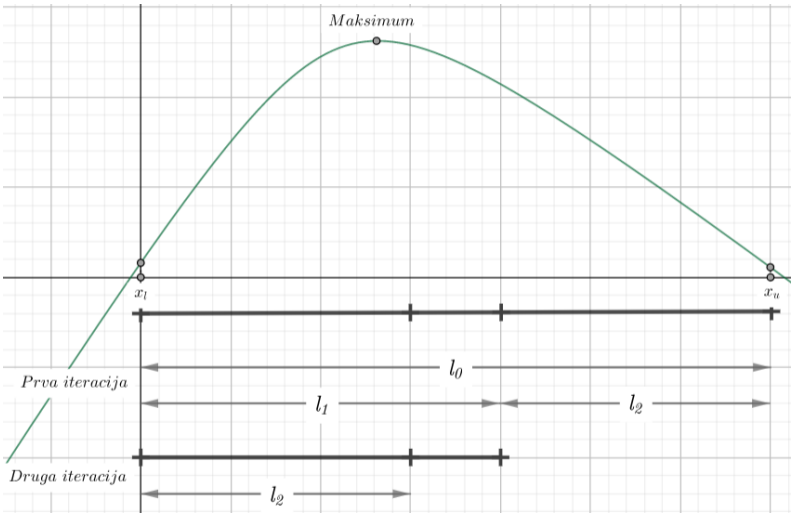
Metoda zlatnog reza slična je metodi bisekcije koja definira početni segment $[x_l, x_u]$ koji zatvara jednu nultočku, odnosno kod metode zlatnog reza zatvara ekstrem. Zbog toga se kaže da metoda zlatnog reza predstavlja algoritam pretraživanja lokalnog ekstrema na zatvorenom intervalu.

Razlika metode zlatnog reza s bisekcijom je da umjesto dvije vrijednosti funkcije, koriste se tri, kako bi se odredilo gdje se nalazi ekstrem. Time se odabire dodatna točka unutar segmenta, što znači da ukupno ima četiri točke. Nakon toga se provjerava na kojem od segmenata $[x_l, x_1]$ i $[x_2, x_u]$ se nalazi ekstrem pri čemu je $x_l < x_2 < x_1 < x_u$.

Ključ ovakvog pristupa je u određivanju unutrašnjih točaka x_1 i x_2. Kao i u bisekciji, cilj je smanjivati evaluaciju funkcije zamjenom starih vrijednosti s novima i to se postiže određivanjem dva uvjeta [7]:

$$l_0 = l_1 + l_2 \qquad \text{(1. uvjet)}$$

$$\frac{l_1}{l_0} = \frac{l_2}{l_1} \qquad \text{(2. uvjet)}$$



Slika 32: Smanjivanje segmeta kod metode Zlatnog reza

Cilj metoda zlatnog reza i bisekcije je da se duljina segmenta na kojemu se nalazi ekstrem konstatno smanjuje nakon svake iteracije, dok će zlatni rez još iskoristiti staru unutrašnju točku za sljedeću iteraciju i tako će se smanjiti broj izvrednjavanja funkcije. Na slici 32 prikazuje se prvi uvjet koji određuje da zbroj l_1 i l_2 mora biti jednak duljini intervala l_0. Drugi uvjet prikazuje omjer duljina novog i starog segmenta za prvu i drugu iteraciju. Za prvu iteraciju na segmentu čija je duljina l_1 nalazi se ekstrem i vrijedi omjer duljina novog i starog segmenta $\frac{l_1}{l_0}$. U drugoj iteraciji odabrani segment iz prve iteracije l_1 imat će duljinu l_2 i u njemu se nalazi ekstrem. Sada je omjer duljina novog i starog segmenta u drugoj iteraciji jednak $\frac{l_2}{l_1}$.

Uvrštavanjem prvog uvjeta u drugi dobivamo

$$\frac{l_1}{l_1 + l_2} = \frac{l_2}{l_1},$$

iz čega uz oznaku $R = \frac{l_2}{l_1}$ slijedi

$$1 + R = \frac{1}{R}$$

odnosno

$$R^2 + R - 1 = 0.$$

Pozitivno rješenje dobivene kvadratne jednadžbe je

$$R = \frac{-1 + \sqrt{1 - 4 \cdot (-1)}}{2} = \frac{\sqrt{5} - 1}{2} = 0.61803\ldots$$

Kao što je predhodno spomenuto metoda započinje s početnim segmentom $[x_l, x_u]$ na kojemu se nalazi samo jedan lokalni ekstrem x_{opt} funkcije f. Nakon toga se računaju dvije unutrašnje točke x_1 i x_2 koje se odabiru prema zlatnom omjeru

$$d = \frac{\sqrt{5} - 1}{2}(x_u - x_l),$$

$$x_1 = x_l + d,\ x_2 = x_u - d.$$

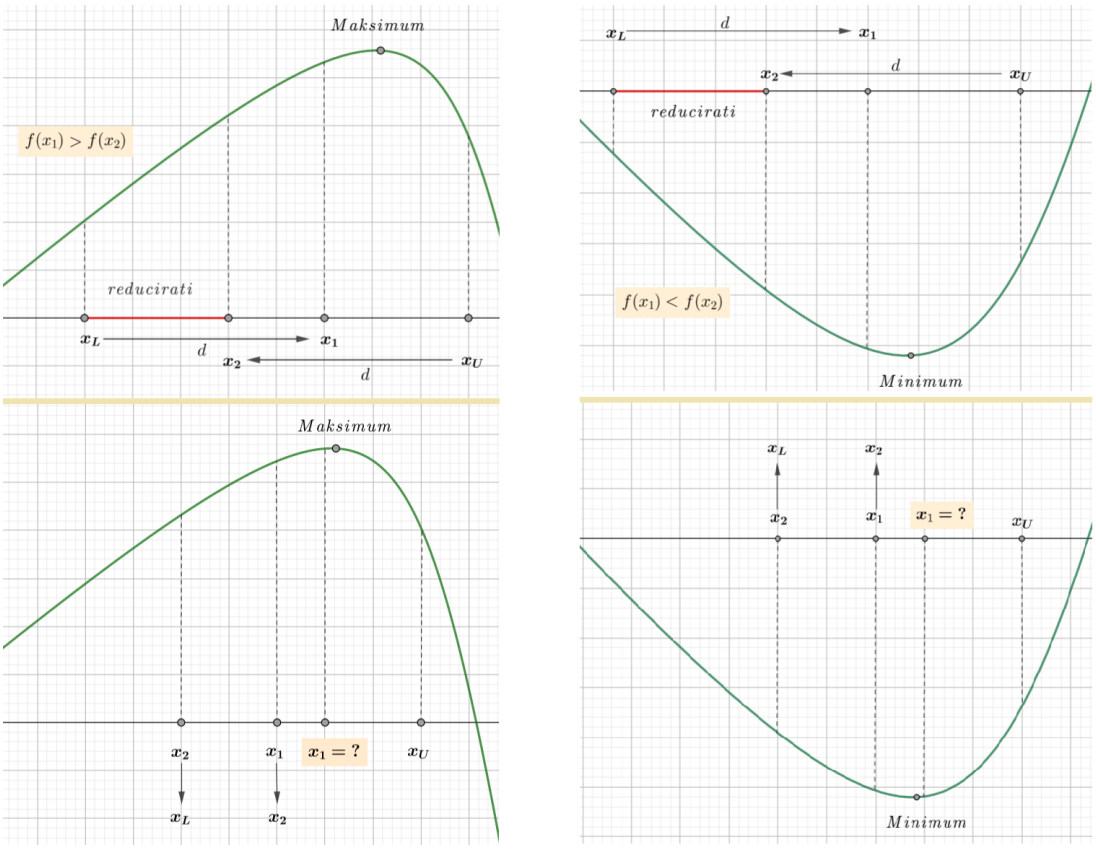
Zatim se funkcija evaluira na tim unutrašnjim točkama i javljaju se dva rezultata. Za traženje minimuma vrijedi:

- Ako je $f(x_1) > f(x_2)$, desni dio intervala s unutrašnjom točkom se eliminira, vrijedi $x_{opt} \in [x_l, x_1]$ i tako x_1 postaje gornja granica x_u za novi segment.
- Ako je $f(x_1) < f(x_2)$, lijevi dio intervala s unutrašnjom točkom se eliminira, vrijedi $x_{opt} \in [x_2, x_u]$ i tako x_2 postaje donja granica x_l za novi segment.

Za traženje maksimuma vrijedi obrnuto:

- Ako je $f(x_1) > f(x_2)$, onda vrijedi $x_{opt} \in [x_2, x_u]$.
- Ako je $f(x_1) < f(x_2)$, onda vrijedi $x_{opt} \in [x_l, x_1]$.

U rijetkim slučajevima, ako je $f(x_1) = f(x_2)$, onda vrijedi $x_{opt} \in [x_1, x_2]$.



Slika 33: Metoda zlatnog reza za traženje lokalnog maksimuma (lijevo) i minimuma (desno)

Na slici 33 prikazuje se postupak redukcije segmenta za pronalazak lokalnih ekstrema. Može se primijetiti da se razlikuju samo po uvjetu za redukciju intervala.

Prednost zlatnog reza je da nakon prve iteracije, ne treba računati sve vrijednosti funkcije jer se vrijednosti nasljeđuju iz prethodne iteracije i zbog toga se računa samo jedna nova vrijednost

$$x_1 = x_l + \frac{\sqrt{5}-1}{2}(x_u - x_l)$$

ili

$$x_2 = x_u - \frac{\sqrt{5}-1}{2}(x_u - x_l).$$

Iteracija se prekida kada je duljina segmenta manja od nekog željenog broja ε, a točnost rješenja može i u pravilu jest veća od željenog broja.

Koraci za primjenu:

1. Odrediti zatvoreni segment $[x_l, x_u]$ za koji se zna da sadrži točno jedan lokalni ekstrem funkcije f.
2. Odrediti unutrašnje točke x_1 i x_2 takve da vrijedi

$$x_1 = x_l + d,$$

$$x_2 = x_u - d,$$

gdje je

$$d = \frac{\sqrt{5}-1}{2}(x_u - x_l).$$

3. Izračunaju se vrijednosti $f(x_1)$ i $f(x_2)$.

4. Za traženje maksimuma:

   - Ako je $f(x_1) < f(x_2)$, tada se nova kalkulacija izvršava samo na x_2 i vrijedi

$$x_u = x_1,$$

$$x_1 = x_2,$$

$$x_2 = x_u - \frac{\sqrt{5}-1}{2}(x_u - x_l).$$

   - Ako je $f(x_1) > f(x_2)$, tada se nova kalkulacija izvršava samo na x_1 i vrijedi

$$x_l = x_2,$$

$$x_2 = x_1,$$

$$x_1 = x_l + \frac{\sqrt{5}-1}{2}(x_u - x_l).$$

   Za traženje minimuma:

   - Ako je $f(x_1) < f(x_2)$, tada se nova kalkulacija izvršava samo na x_1 i vrijedi

$$x_l = x_2,$$

$$x_2 = x_1,$$

$$x_1 = x_l + \frac{\sqrt{5}-1}{2}(x_u - x_l).$$

   - Ako je $f(x_1) > f(x_2)$, tada se nova kalkulacija izvršava samo na x_2 i vrijedi

$$x_u = x_1,$$

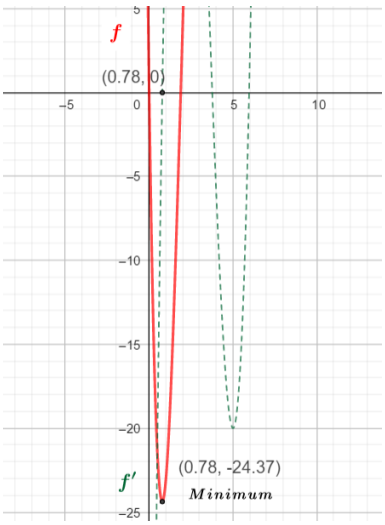
$$x_1 = x_2,$$

$$x_2 = x_u - \frac{\sqrt{5}-1}{2}(x_u - x_l).$$

5. Iteracija se zaustavlja kada se ispuni uvjet $x_u - x_l < \varepsilon$.

Primjer 8. Pomoću metode zlatnog reza pronađite lokalni minimum funkcije $f(x) = x^4 - 14x^3 + 60x^2 - 70x$ na segmentu $[0.5, 1.5]$ s točnošću $\varepsilon = 0.01$ [3].

n	x_l	x_u	x_1	x_2	$f(x_1)$	$f(x_2)$
0	0.5	1.5	1.1180	0.8820	−21.2655	−24.0654
1	0.5	1.1180	0.8820	0.7361	−24.0654	−24.3066
2	0.5	0.8820	0.7361	0.6459	−24.3066	−23.7802
3	0.6459	0.8820	0.7918	0.7361	−24.3659	−24.3066
4	0.7361	0.8820	0.8262	0.7918	−24.3071	−24.3659
5	0.7361	0.8262	0.7918	0.7705	−24.3659	−24.3663
6	0.7361	0.7918	0.7705	0.7574	−24.3663	−24.3524
7	0.7574	0.7918	0.7786	0.7705	−24.3694	−23.3663
8	0.7705	0.7918	0.7837	0.7786	−24.3694	−24.3694
9	0.7705	0.7837	0.7786	0.7755	−24.3694	−24.3687
10	0.7755	0.7837	0.7806	0.7786	−24.3696	−24.3694



Tablica 6: Prikaz metode zlatnog reda za lokalni minimum za funkciju $f(x) = x^4 - 14x^3 + 60x^2 - 70x$ i njezin graf

Iz tablice 6 vidi se da se uvjet zaustavljanja ispunio u 10 iteracija gdje $x_{opt} = 0.7805598$ i taj lokalni minimum se može potvrditi i na grafu funkcije za segment $[0.5, 1.5]$.

Za isti primjer promjenom segmenta može se naći lokalni maksimum tako da sada početni segment bude $[3.5, 4]$.

n	x_l	x_u	x_1	x_2	$f(x_1)$	$f(x_2)$
0	3.5	4	3.8090	3.6910	40.6956	40.6583
1	3.6910	4	3.8820	3.8090	40.5379	40.6956
2	3.6910	3.8820	3.8090	3.7639	40.6956	40.7245
3	3.6910	3.8090	3.7639	3.7361	40.7245	40.7158
4	3.7361	3.8090	3.7812	3.7639	40.7197	40.7245
5	3.7361	3.7812	3.7639	3.7533	40.7245	40.7235
6	3.7533	3.7812	3.7705	3.7639	40.7236	40.7245
7	3.7533	3.7705	3.7639	3.7599	40.72447	40.72446
8	3.7599	3.7705	3.7664	3.7639	40.7243	40.7245
9	3.7599	3.7664	3.7639	3.7624	40.72447	40.72452

Tablica 7: Prikaz metode zlatnog reda za lokalni maksimum za funkciju $f(x) = x^4 - 14x^3 + 60x^2 - 70x$ i njezin graf

Iz tablice 7 vidi se da se uvjet zaustavljanja ispunio u 9 iteracija gdje $x_{opt} = 3.7623792$ i taj lokalni maksimum se može potvrditi i na grafu funkcije za segment $[3.5, 4]$.

3.2.2. Fibonaccijeva metoda

Fibonaccijeva metoda uvjetuje da funkcija na zadanom segmentu ima jedan lokalni ekstrem i predstavlja algoritam za pronalazak istog određivanjem unutrašnje točke za redukciju segmenta. Svakim korakom, odnosno iteracijom, odabiru se intervali tako da se uvijek računa samo jedna nova točka, ovisno o uvjetu. Iz toga se vidi da Fibonaccijeva metoda ima sličnosti s metodom zlatnog reza, a glavna razlika je u koeficijentu redukcije. Kod metode zlatnog reza koeficijent redukcije je uvijek isti, a u Fibonaccijevoj metodi se mijenja kod svake iteracije i u k-toj iteraciji iznosi $\frac{F_{n-k}}{F_{n-k+1}}$.

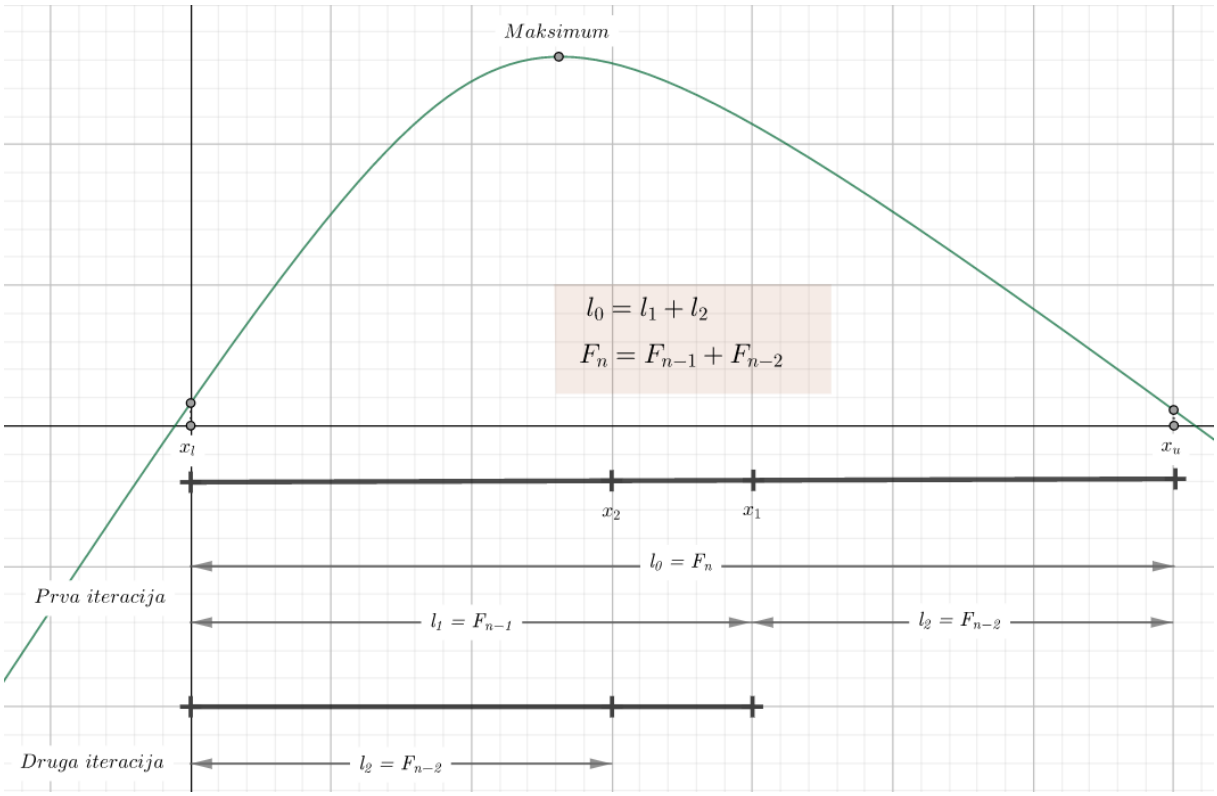
Definicija 8. *Fibonaccijeva metoda se bazira na Fibonaccijevom nizu koji se definira rekurzivno s*

$$F_0 = 0,\ F_1 = 1,\ F_n = F_{n-1} + F_{n-2},\ \ n = 2, 3, 4, \ldots$$

Prema tome Fibonaccijevi brojevi su $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}
0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987

Tablica 8: Fibonaccijev niz



Slika 34: Redukcija segmenta kod Fibonaccijeve metode

Duljina segmenta smanjuje se s obzirom na Fibonaccijeve brojeve. Omjer duljina novog i starog segmenta u prvoj iteraciji ima faktor redukcije $\frac{F_{n-1}}{F_n}$.

$$\frac{l_1}{l_0} = \frac{F_{n-1}}{F_n}$$

$$l_0 = l_1 + l_2$$
$$1 = \frac{l_1}{l_0} + \frac{l_2}{l_0}$$
$$1 = \frac{F_{n-1}}{F_n} + \frac{l_2}{l_0}$$
$$\frac{l_2}{l_0} = 1 - \frac{F_{n-1}}{F_n}$$
$$\frac{l_2}{l_0} = \frac{F_n - F_{n-1}}{F_n}$$
$$\frac{l_2}{l_0} = \frac{F_{n-2}}{F_n}.$$

Stoga je u drugoj iteraciji omjer segmenata jednak

$$\frac{l_2}{l_1} = \frac{l_2}{l_0} \cdot \frac{l_0}{l_1}$$

$$\frac{l_2}{l_1} = \frac{F_{n-2}}{F_n} \cdot \frac{F_n}{F_{n-1}}$$

$$\frac{l_2}{l_1} = \frac{F_{n-2}}{F_{n-1}}.$$

Nakon k-te iteracije induktivno slijedi da je

$$\frac{l_k}{l_{k-1}} = \frac{F_{n-k}}{F_{n-k+1}}.$$

Za Fibonaccijevu metodu potrebno je znati koliki će biti broj podintervala, odnosno iteracija unutar tražene točnosti ε. Duljina zadnjeg segmenta je

$$\frac{F_{n-1}}{F_n} \cdot \frac{F_{n-2}}{F_{n-1}} \cdots \frac{F_3}{F_4} \cdot \left(\frac{F_2}{F_3} + \delta\right) \cdot (x_u - x_l) = \frac{1 + 2\delta}{F_n}(x_u - x_l)$$

pri čemu je $\delta > 0$ dovoljno mali odabrani broj. Stoga mora biti

$$\frac{1 + 2\delta}{F_n}(x_u - x_l) < \varepsilon$$

ili

$$F_n > \frac{1 + 2\delta}{\varepsilon}(x_u - x_l).$$

Unutrašnje točke x_1 i x_2 segmenta $[x_l, x_u]$ k-te iteracije računaju se prema formulama

$$x_1 = x_l + \frac{F_{n-k}}{F_{n-k+1}}(x_u - x_l),$$

$$x_2 = x_l + \frac{F_{n-k-1}}{F_{n-k+1}}(x_u - x_l)$$

ili

$$x_2 = x_u - \frac{F_{n-k}}{F_{n-k+1}}(x_u - x_l).$$

Nakon što se odrede točke x_1 i x_2 postupak traženja lokalnog ekstrema se dalje nastavlja na isti način kao i kod metode zlatnog reza kako je objašnjeno na stranici 38.

Problem ove metode je u $n-2$ iteraciji kada se javlja faktor redukcije $\frac{F_2}{F_3} = \frac{1}{2}$. Tada se dvije središnje točke podudaraju u sredini reduciranog segmenta pa se segment ne može dalje reducirati. Stoga u toj iteraciji unutrašnje točke računamo prema formulama

$$\begin{aligned} x_1 &= x_l + \left(\tfrac{1}{2} + \delta\right)(x_u - x_l), \\ x_2 &= x_l + \left(\tfrac{1}{2} - \delta\right)(x_u - x_l). \end{aligned} \qquad (\clubsuit)$$

U implementaciji Fibonaccijeve metode je odabrano $\delta = 0.01$.

Primjer 9. Pomoću Fibonaccijeve metode pronađite lokalni minimum funkcije $f(x) = x^4 - 14x^3 + 60x^2 - 70x$ na segmentu $[0.5, 1.5]$ s točnošću $\varepsilon = 0.01$, kao iz prošlog poglavlja.

Za traženu točnost računa se koliko će iteracija biti potrebno i ukupan broj iteracija N čita se iz tablice 8:

$$F_n > \frac{1 + 2\delta}{\varepsilon}(x_u - x_l)$$

$$F_n > \frac{1 + 0.02}{0.01}(1.5 - 0.5) = 102$$

$$n = 12$$

k	x_l	x_u	x_1	x_2	$f(x_1)$	$f(x_2)$
0	0.5	1.5	1.1181	0.8819	−21.2651	−24.0655
1	0.5	1.1181	0.8819	0.7361	−24.0655	−24.3068
2	0.5	0.8819	0.7361	0.6458	−24.3068	−23.7796
3	0.6458	0.8819	0.7917	0.7361	−24.3660	−24.3068
4	0.7361	0.8819	0.8264	0.7917	−24.3067	−24.3660
5	0.7361	0.8264	0.7917	0.7708	−24.3660	−24.3665
6	0.7361	0.7917	0.7708	0.7569	−24.3665	−24.3518
7	0.7569	0.7917	0.7778	0.7708	−24.3693	−24.3665
8	0.7708	0.7917	0.7847	0.7778	−24.3691	−24.3693
9	0.7708	0.7847	0.7779	0.7776	−24.36932	−24.36927
10	0.7776	0.7847	−	−	−	−

Tablica 9: Prikaz Fibonaccijeve metode za lokalni minimum za funkciju $f(x) = x^4 - 14x^3 + 60x^2 - 70x$

Za $k = 9$ u tablici 9 javlja se kvocijent $\frac{F_2}{F_3}$ pa se unutrašnje točke računaju prema formulama (♣). Kako je u ovom slučaju $f(x_2) > f(x_1)$ u sljedećem koraku $k = 10$ je $x_l = x_2 = 0.7776$, x_u ostaje nepromijenjen pa je $x_{opt} = 0.5 \cdot (x_u + x_l) = 0.78118$.

Na istom primjeru kao i u predhodnom poglavlju može se naći lokalni maksimum s početnim segmentom $[3.5, 4]$.

Također, računa se koliko će iteracija biti potrebno:

$$F_n > \frac{1 + 2\delta}{\varepsilon}(x_u - x_l)$$

$$F_n > \frac{1 + 0.02}{0.01}(4 - 3.5) = 51$$

$$n = 10$$

k	x_l	x_u	x_1	x_2	$f(x_1)$	$f(x_2)$
0	3.5	4	3.8091	3.6909	40.6955	40.6582
1	3.6909	4	3.8818	3.8091	40.5384	40.6955
2	3.6909	3.8818	3.8091	3.7636	40.6955	40.7245
3	3.6909	3.8091	3.7636	3.7364	40.7245	40.7160
4	3.7364	3.8091	3.7818	3.7636	40.7193	40.7245
5	3.7364	3.7818	3.7636	3.7545	40.7245	40.7238
6	3.7545	3.7818	3.7727	3.7636	40.7230	40.7245
7	3.7545	3.7727	3.7638	3.7635	40.72447	40.72449
8	3.7545	3.7638	–	–	–	–

Tablica 10: Prikaz Fibonaccijeve metode za lokalni maksimum za funkciju $f(x) = x^4 - 14x^3 + 60x^2 - 70x$

Za $k = 7$ u tablici 10 javlja se kvocijent $\frac{F_2}{F_3}$ pa se unutrašnje točke računaju prema formulama (♣). Kako je u ovom slučaju $f(x_2) > f(x_1)$ u sljedećem koraku $k = 8$ je $x_u = x_1 = 3.7638$, x_l ostaje nepromijenjen pa je $x_{opt} = 0.5 \cdot (x_u + x_l) = 3.75918$.

U metodi zlatnog reza, kod traženja lokalnog maksimuma, segment posljednje iteracije je $[3.7599, 3.7664]$ duljine 0.0065, a kod Fibonaccijeve metode je $[3.7545, 3.7638]$ duljine 0.0093 s tim da je lokalni maksimum u Fibonaccijevoj metodi dobiven u 8 iteracija dok je u metodi zlatnog reza trebalo 9 iteracija. Kod traženja lokalnog minimuma, u metodi zlatnog reza segment posljednje iteracije je $[0.7755, 0.7837]$ duljine 0.0082, a kod Fibonaccijeve metode je $[0.7776, 0.7847]$ duljine 0.0071 s tim da je lokalni minimum dobiven u 10 iteracija kod obje metode. Iz toga može se zaključiti da primjenom Fibonaccijeve metode dobivamo manji segment unutar kojeg se nalazi lokalni ekstrem nego što je to slučaj kod metode zlatnog reza kada imamo isti broj iteracija.

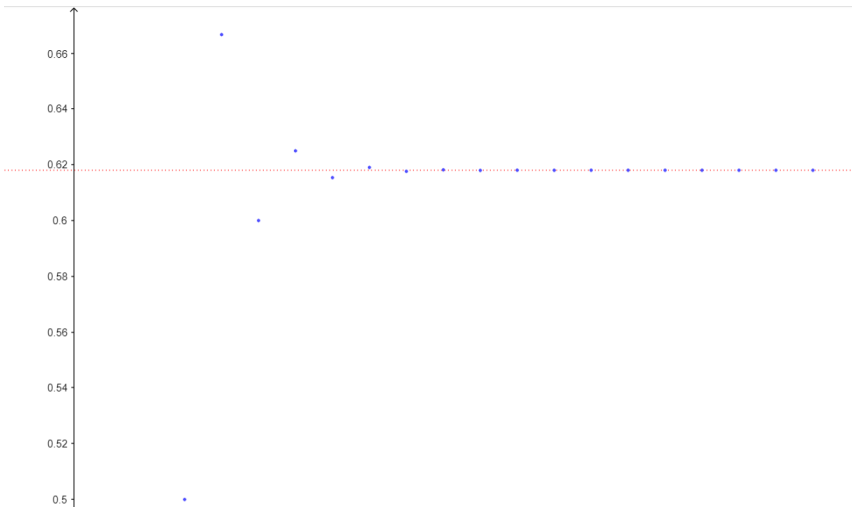
3.2.3. Prednosti i nedostaci direktnih metoda

Jedna od razlika metode zlatnog reza i Fibonaccijeve metode je u faktoru redukcije. Faktor redukcije kod metode zlatnog reza je konstantan, dok se kod Fibonaccijeve metode svakom iteracijom mijenja.
Niz kvocijenata dvaju uzastopnih članova Fibonaccijevog niza konvergira prema konstanti zlatnog reza, tj. vrijedi

$$\lim_{n\to\infty} \frac{F_{n-1}}{F_n} = \frac{\sqrt{5}-1}{2} \approx 0.61803\cdots$$

n	$\frac{F_{n-1}}{F_n}$	R
3	0.5	0.6180339887498949
4	0.6666666666666666	0.6180339887498949
5	0.6	0.6180339887498949
6	0.625	0.6180339887498949
7	0.6153846153846154	0.6180339887498949
8	0.6190476190476191	0.6180339887498949
9	0.6176470588235294	0.6180339887498949
10	0.6181818181818182	0.6180339887498949
11	0.6179775280898876	0.6180339887498949
12	0.6180555555555556	0.6180339887498949
13	0.6180257510729614	0.6180339887498949
14	0.6180371352785146	0.6180339887498949
15	0.6180327868852459	0.6180339887498949
16	0.6180344478216818	0.6180339887498949
17	0.6180338134001252	0.6180339887498949
18	0.6180340557275542	0.6180339887498949
19	0.6180339631667066	0.6180339887498949
20	0.6180339985218034	0.6180339887498949

Tablica 11: Prikaz konvergencije članova Fibonaccijevog niza prema konstanti zlatnog reza



Slika 35: Grafički prikaz konvergencije Fibonaccijevog niza prema konstanti zlatnog reza

Iz tablice 11 i slike 35 se vidi da je ta konvergencija brza.

Druga razlika je da broj iteracija kod Fibonaccijeve metode treba biti poznat i određen prema traženoj točnosti prije početka pokretanja algoritma. To je jedan od glavnih nedostataka Fibonaccijeve metode jer se kod metode zlatnog reza iteracije izvode do željene ili postavljene točnosti, iz čega reducirani intervali nisu ovisni o broju iteracija i zbog toga metoda zlatnog reza je jednostavnija metoda za uporabu.
Ako je broj iteracija isti, reducirani segment u metodi zlatnog reza je veći od onog u Fibonaccijevoj metodi. Za postizanje zadane točnosti metoda zlatnog reza treba više iteracija od Fibonaccijeve metode. Odnos duljina zadnjih intervala kod metode zlatnog reza i Fibonaccijeve metode dokazuje veću preciznost Fibonaccijeve metode.
Da bi iskoristili točno n iteracija kod Fibonaccijeve metode kreće se od $(n+2)$-og Fibonaccijevog člana i duljina zadnjeg intervala je

$$\frac{1+2\delta}{F_{n+2}}(x_u - x_l),$$

a duljina zadnjeg intervala kod metode zlatnog reza je

$$R^n(x_u - x_l).$$

Konačno, promatramo kvocjent

$$\frac{R^n(x_u - x_l)}{\frac{1+2\delta}{F_{n+2}}(x_u - x_l)} = \frac{R^n}{\frac{1+2\delta}{F_{n+2}}} = \frac{R^n F_{n+2}}{1+2\delta}$$

koji je za dovoljno veliki broj iteracija približno jednak 1.15 ako uzmemo $\delta = 0.01$.

n	$\frac{1.02}{F_{n+2}}$	R^n	$R^n \Big/ \frac{1.02}{F_{n+2}}$
1	0.51	0.6180339887498949	1.21183135048999
2	0.34	0.3819660112501052	1.1234294448532505
3	0.20400000000000001	0.23606797749978975	1.157195968136224
4	0.1275	0.14589803375031551	1.1442983039240433
5	0.07846153846153846	0.09016994374947428	1.1492247732776133
6	0.04857142857142857	0.055728090000841245	1.1473430294290843
7	0.03	0.034441853748633046	1.1480617916211013
8	0.018545454545454546	0.021286236252208202	1.1477872488935796
9	0.01146067415730337	0.013155617496424849	1.147892114884129
10	0.007083333333333334	0.008130618755783355	1.147852059640003
11	0.004377682403433477	0.005024998740641495	1.1478673593818316
12	0.0027055702917771884	0.003105620015141861 6	1.1478615154004723
13	0.0016721311475409836	0.001919378725499634	1.1478637476027223
14	0.0010334346504559271	0.0011862412896422284	1.1478628949773326
15	0.000638697557921102	0.0007331374358574057	1.1478632206512518
16	0.0003947368421052631 5	0.0004531038537848227 4	1.1478630962548841
17	0.0002439607749342262 6	0.0002800335820725831	1.1478631437700686
18	0.0001507760532150776 2	0.0001730702717122396 2	1.1478631256208833
19	$9.318472501370364 \cdot 10^{-5}$	0.0001069633103603435 6	1.1478631325532553
20	$5.759132742363503 \cdot 10^{-5}$	$6.610696135189609_{,}dot10^{-5}$	1.147863129905325

Tablica 12: Odnos duljina zadnjih intervala kod metode zlatnog reza i Fibonaccijeve metode

Željena točnost za rješavanje minimizacijskog problema često neće biti navedena u smislu određenog broja iteracija, zbog čega je to još jedan nedostatak Fibonaccijeve metode. Međutim, za mali broj iteracija Fibonaccijeva metoda je efikasnija od metode zlatnog reza.

3.2.4. Implementacija direktnih metoda u Octave programskom jeziku

```
## [x] = fibonacci_min(0,1,@func_x,16)
% Fibonaccijeva metoda za trazenje lokalnog MINIMUMA
function x = fibonacci_min(xL, xU, f, iter)
  n = iter + 2;
  fib(1) = 1;   % F1=1
  fib(2) = 1;   % F2=1
  for jj = 3 : n
    fib(jj) = fib(jj - 1) + fib(jj - 2);  % formula for other Fibonacci
    numbers
  end

  if (n == 3) % samo jedna iteracija
    x1 = xL + (fib(n - 1) / fib(n) + D) * (xU - xL);
    x2 = xL + (fib(n - 2) / fib(n) - D) * (xU - xL);
  else
    x1 = xL + fib(n - 1) / fib(n) * (xU - xL);
    x2 = xL + fib(n - 2) / fib(n) * (xU - xL);
  endif

  f_x1 = f(x1);
  f_x2 = f(x2);

  for i = 2 : n - 1
    % display iteartions
    disp(strcat('xL=', num2str(xL,10),
                ', xU=', num2str(xU,10),
                ', x1=', num2str(x1,10),
                ', x2=',num2str(x2,10),
                ', f(x1)=',num2str(f_x1,10),
                ', f(x2)=',num2str(f_x2,10),
                ', xU-xL=',num2str(xU-xL,10)))

    if(f_x1 < f_x2)                                    % za max f_x1 > f_x2
      xL = x2;
      if (i == n - 2)
        x2 = xL + 0.49 * (xU - xL);
        x1 = xL + 0.51 * (xU - xL);
        f_x2 = f(x2);
      else
        x2 = x1;
        x1 = xL + fib(n - i) / fib(n - i + 1) * (xU - xL);
        f_x2 = f_x1;
      endif
      f_x1 = f(x1);
```

```
44      else
45        xU = x1;
46        if (i == n - 2)
47          x2 = xL + 0.49 * (xU - xL);
48          x1 = xL + 0.51 * (xU - xL);
49          f_x1 = f(x1);
50        else
51          x1 = x2;
52          x2 = xU - fib(n - i) / fib(n - i + 1) * (xU - xL);
53          f_x1 = f_x2;
54        endif
55        f_x2 = f(x2);
56      endif
57
58    endfor
59
60    x = 0.5 * (xL + xU);
61
62  % ispis zadnjeg koraka za i = n
63  disp(strcat('xL=', num2str(xL,10),
64              ', xU=', num2str(xU,10),
65              ', x=', num2str(x,10),
66              ', f(x)=',num2str(f(x),10),
67              ', xU-xL=',num2str(xU-xL,10)))
```

Algoritam 3.9: Octave kod Fibonaccijevu metodu za traženje lokalnog minimuma

Navedeni Octave kod možemo primijeniti i za traženje lokalnog maksimuma što zahtijeva zamjenu predanog uvjeta (linija 32) s odgovarajućim uvjetom što je prikazano u kodu.

```
1  ## [x] = golden_section_min(0, 1, @func_x, 0.0001)
2  % Metoda Zlatnog reza za trazenje lokalnog MINIMUMA
3  function [x, itCount] = golden_section_min(xL, xU, f, epsilon, iter=100)
4    R =  ((sqrt(5) - 1) / 2);   % golden ratio = 0.618
5    d = R * (xU - xL);
6    x1 = xL + d;
7    x2 = xU - d;
8    f_x1 = f(x1);
9    f_x2 = f(x2);
10   if(f_x1 < f_x2)                                  % za max f_x1 > f_x2
11       x  = x1;
12     else
13       x  = x2;
14     endif
15
16   for itCount = 0 : iter
17     % display iteartions
18     disp(strcat('xL=', num2str(xL,10),
19                 ', xU=', num2str(xU,10),
20                 ', x1=', num2str(x1,10),
21                 ', x2=',num2str(x2,10),
22                 ', f(x1)=',num2str(f_x1,10),
23                 ', f(x2)=',num2str(f_x2,10),
24                 ', xU-xL=',num2str(xU-xL)))
25     if (abs (xU-xL) < epsilon)
26       printf('\nbroj iteracija = %i\n', itCount);
27       return;
28     endif
29
30     d = R * d;
31     if(f_x1 < f_x2)                                  % za max f_x1 > f_x2
32       xL = x2;
33       x2 = x1;
34       x1 = xL + d;
35       x  = x1;
36       f_x2 = f_x1;
37       f_x1 = f(x1);
38     else
39       xU = x1;
40       x1 = x2;
41       x2 = xU - d;
42       x  = x2;
43       f_x1 = f_x2;
44       f_x2 = f(x2);
45     endif
46
47   endfor
48   error('metoda zlatnog reza nije uspjela sa %i iteracija.', iter)
```

Algoritam 3.10: Octave kod metode zlatnog reza za traženje lokalnog minimuma

Isto kao i kod Fibonaccijeve metode, Octave kod možemo primijeniti za traženje lokalnog maksimuma što zahtijeva zamjenu predanog uvjeta (linije 10 i 32) kako je prikazano u komentarima.

Primjer 10. Traži se lokalni minimum funkcije $f(x) = x^2 - \sin x$ na intervalu $[0, 1]$ s Fibonaccijevom metodom i točnošću manjom od $\varepsilon = 10^{-3}$ [10].

```
>> fibonacci_min(0, 1, @func_x, 15)
xL=0, xU=1, x1=0.6180338134, x2=0.3819661866, f(x1)=-0.197468007, f(x2)=-0.2268475115, xU-xL=1
xL=0, xU=0.6180338134, x1=0.3819661866, x2=0.2360676268, f(x1)=-0.2268475115, f(x2)=-0.1781532106, xU-xL=0.6180338134
xL=0.2360676268, xU=0.6180338134, x1=0.4721352536, x2=0.3819661866, f(x1)=-0.2318772782, f(x2)=-0.2268475115, xU-xL=0.3819661866
xL=0.3819661866, xU=0.6180338134, x1=0.5278647464, x2=0.4721352536, f(x1)=-0.2250486878, f(x2)=-0.2318772782, xU-xL=0.2360676268
xL=0.3819661866, xU=0.5278647464, x1=0.4721352536, x2=0.4376956794, f(x1)=-0.2318772782, f(x2)=-0.2322759904, xU-xL=0.1458985598
xL=0.3819661866, xU=0.4721352536, x1=0.4376956794, x2=0.4164057608, f(x1)=-0.2322759904, f(x2)=-0.2310822086, xU-xL=0.090169067
xL=0.4164057608, xU=0.4721352536, x1=0.450845335, x2=0.4376956794, f(x1)=-0.232465042, f(x2)=-0.2322759904, xU-xL=0.0557294928
xL=0.4376956794, xU=0.4721352536, x1=0.458985598, x2=0.450845335, f(x1)=-0.2323711421, f(x2)=-0.232465042, xU-xL=0.0344395742
xL=0.4376956794, xU=0.458985598, x1=0.450845335, x2=0.4458359424, f(x1)=-0.232465042, f(x2)=-0.2324425728, xU-xL=0.0212899186
xL=0.4458359424, xU=0.458985598, x1=0.4539762054, x2=0.450845335, f(x1)=-0.2324480538, f(x2)=-0.232465042, xU-xL=0.0131496556
xL=0.4458359424, xU=0.4539762054, x1=0.450845335, x2=0.4489668128, f(x1)=-0.232465042, f(x2)=-0.2324637727, xU-xL=0.008140262993
xL=0.4489668128, xU=0.4539762054, x1=0.4520976832, x2=0.450845335, f(x1)=-0.2324611133, f(x2)=-0.232465042, xU-xL=0.005009392611
xL=0.4489668128, xU=0.4520976832, x1=0.450845335, x2=0.4502191609, f(x1)=-0.232465042, f(x2)=-0.2324655736, xU-xL=0.003130870382
xL=0.4489668128, xU=0.450845335, x1=0.4502191609, x2=0.4495929869, f(x1)=-0.2324655736, f(x2)=-0.2324651505, xU-xL=0.001878522229
xL=0.4495929869, xU=0.450845335, x1=0.4502316844, x2=0.4502066374, f(x1)=-0.2324655723, f(x2)=-0.2324655745, xU-xL=0.001252348153
xL=0.4495929869, xU=0.4502316844, x=0.4499123356, f(x)=-0.2324654856, xU-xL=0.0006386975579
ans =    4.499123356293049e-01
```

Slika 36: Pozivanje funkcije `fibonacci_min` i izlazne vrijednosti

Iz slike 36 broj iteracija je unaprijed računat prema formuli

$$F_n > \frac{1 + 2\delta}{\varepsilon}(x_u - x_l) > 1020,\ n = 17$$

iz čega slijedi $x_{opt} = 0.449912$.
Poziv funkcije `golden_section_min` iskoristi će se s vrijednostima iz primjera 10 kako bi se usporedili dobiveni rezultati.

```
>> golden_section_min(0, 1, @func_x, 0.001)
xL=0, xU=1, x1=0.6180339887, x2=0.3819660113, f(x1)=-0.1974679332, f(x2)=-0.2268474828, xU-xL=1
xL=0, xU=0.6180339887, x1=0.3819660113, x2=0.2360679775, f(x1)=-0.2268474828, f(x2)=-0.178153386, xU-xL=0.61803
xL=0.2360679775, xU=0.6180339887, x1=0.472135955, x2=0.3819660113, f(x1)=-0.2318772406, f(x2)=-0.2268474828, xU-xL=0.38197
xL=0.3819660113, xU=0.6180339887, x1=0.527864045, x2=0.472135955, f(x1)=-0.2250488223, f(x2)=-0.2318772406, xU-xL=0.23607
xL=0.3819660113, xU=0.527864045, x1=0.472135955, x2=0.4376941013, f(x1)=-0.2318772406, f(x2)=-0.2322759425, xU-xL=0.1459
xL=0.3819660113, xU=0.472135955, x1=0.4376941013, x2=0.416407865, f(x1)=-0.2322759425, f(x2)=-0.2310823806, xU-xL=0.09017
xL=0.416407865, xU=0.472135955, x1=0.4508497187, x2=0.4376941013, f(x1)=-0.2324650349, f(x2)=-0.2322759425, xU-xL=0.055728
xL=0.4376941013, xU=0.472135955, x1=0.4589803375, x2=0.4508497187, f(x1)=-0.232371255, f(x2)=-0.2324650349, xU-xL=0.034442
xL=0.4376941013, xU=0.4589803375, x1=0.4508497187, x2=0.44582472, f(x1)=-0.2324650349, f(x2)=-0.2324424539, xU-xL=0.021286
xL=0.44582472, xU=0.4589803375, x1=0.4539553388, x2=0.4508497187, f(x1)=-0.2324482461, f(x2)=-0.2324650349, xU-xL=0.013156
xL=0.44582472, xU=0.4539553388, x1=0.4508497187, x2=0.44893034, f(x1)=-0.2324650349, f(x2)=-0.232463663, xU-xL=0.0081306
xL=0.44893034, xU=0.4539553388, x1=0.45203596, x2=0.4508497187, f(x1)=-0.2324613965, f(x2)=-0.2324650349, xU-xL=0.005025
xL=0.44893034, xU=0.45203596, x1=0.4508497187, x2=0.4501165813, f(x1)=-0.2324650349, f(x2)=-0.2324655697, xU-xL=0.0031056
xL=0.44893034, xU=0.4508497187, x1=0.4501165813, x2=0.4496634775, f(x1)=-0.2324655697, f(x2)=-0.2324652458, xU-xL=0.0019194
xL=0.4496634775, xU=0.4508497187, x1=0.4503966149, x2=0.4501165813, f(x1)=-0.2324655199, f(x2)=-0.2324655697, xU-xL=0.0011862
xL=0.4496634775, xU=0.4503966149, x1=0.4501165813, x2=0.449943511, f(x1)=-0.2324655697, f(x2)=-0.232465505, xU-xL=0.00073314

broj iteracija = 15
ans =    4.499435110398016e-01
```

Slika 37: Pozivanje funkcije `golden_section_min` i izlazne vrijednosti

Primjećuje se da je do zadane točnosti bilo potrebno 15 iteracija, gdje je $x_{opt} = 0.449944$.

Oba rezultata dala su lokalni ekstrem reduciranjem segmenta kroz 15 iteracija. Razlika koju možemo primijetiti je da je duljina $x_u - x_l$ zadnjeg segmenta kod Fibonaccijeve metode manja nego što je kod metode zlatnog reza.

4. Zaključak

U ovom radu prikazane su različite metode za rješavanja problema optimizacije. Kao što se općenito zna, funkcije predstavljaju različite situacije u kojoj se traži najveća ili najmanja vrijednost tih situacija, odnosno u matematici to su ekstremi funkcija. Najjednostavniji način pronalaska ekstrema je uz pomoć derivacija, a u nekim drugim situacijama koriste se numeričke metode koje su pogodne za rješavanje vrlo složenih optimizacijskih problema.

Numerička matematika koristi se za određivanje približnih rješenja optimizacijskih problema koja koristi iteracijske postupke, a cilj je u što manje koraka doći do što preciznijeg rješenja. Za svaki problem postoji metoda koja je učinkovitija od drugih metoda, zbog čega je bitno znati značajke svake kako bi se znalo koju koristiti. Pošto numeričke metode koriste iterativne postupke, najlakše ih je prikazati putem računala koji puno brže provodi takve postupke.

Primjena na računalu prikazana je uz pomoć Octave softvera kako bi se razlike brže uočile u korištenju, a sve slike koje su prikazane za lakše razumijevanje izrađene su u GeoGebri.

Popis literature

[1] A. Antoniou i W.-S. Lu, „Practical Optimization: Algorithms Engineering Aplications," Department of Electrical i Computer Engineering University of Victoria, Canada, 2007., str. 1–4, 81–92, 579–580.

[2] T. Hunjak, „Matematika- Ekonomika poduzetništva," str. 1304–1647, 2013.

[3] W. Yin, „Math 273a," *Optimization 1D search methods*, 2015.

[4] D. Zlatko, M. Miljenko, S. Sanja, H. Vjeran, R. Mladen i S. Saša, „Numerička analiza," Sveučilište u Zagrebu PMF – Matematički odjel, 2003., str. 449–454, 460–466, 475–476, 580–583.

[5] M. Sussman, *Roots of Equations*, 2016. adresa: `http://www.math.pitt.edu/%5C~sussmanm/2070/lab%5C_03/index.html`.

[6] ——, *Newton's method*, 2016. adresa: `www.math.pitt.edu/%5C~sussmanm/2070/lab%5C_04/index.html%5C#Introduction`.

[7] S. C. Chapra i R. P.Canale, „Numerical Methods for Engineers, Seventh Edition," McGraw-Hill Education, 2015., str. 355–363.

[8] D. H. Xu, „MATH3016," *Optimization*, str. 4–17,

[9] *Fibonacci Numbers and Golden Ratio*. adresa: `https://www.onlinemathlearning.com/fibonacci-numbers.html`.

[10] K. D. F. John H. Mathews, „Numerical Method using Matlab, 4th Edition," *The Fibonacci Search*, 2004.

Popis slika

Popis tablica