Prirodno-matematički fakultet, Univerzitet u Nišu, Srbija
http://www.pmf.ni.ac.rs/mii
Matematika i informatika 3 (2) (2016), 30-79

Iterativni algoritmi za rešavanje nelinearnih jednačina i sistema

Mihailo Krstić
Student matematike, Prirodno-matematički fakultet Niš
mihailo1994@yahoo.com

1 Uvod

Problem rešavanja jednačina je jedan od najstarijih problema u matematici. Poznato je da u opštem slučaju nije moguće izraziti korene polinoma, stepena većeg od 4, preko svojih koeficijenata i operacija sabiranja, množenja i korenovanja. Takođe, one jednačine koje opisiju neku fizičku veličinu uglavnom nije moguće rešiti analitički, ili ako je to moguće najčešće rešenje je analitički vrlo glomazno. Na primer, Kardanove[1] formule daju analitičke izraze za rešenja kubne jednačine ali su ona toliko analitički komplikovana da se u praksi ne koriste. Na primer, jednostavna jednačina $x^3 - x^2 + x + 5 = 0$ u skupu $\mathbb{C}$ ima rešenja

$$x_1 = \frac{1}{3}\left(1 - \frac{2}{\sqrt[3]{3\sqrt{561} - 71}} + \sqrt[3]{3\sqrt{561} - 71}\right),$$

$$x_2 = \frac{1}{3} + \frac{1 + i\sqrt{3}}{3\sqrt[3]{3\sqrt{561} - 71}} - \frac{1}{6}\left(1 - i\sqrt{3}\right)\sqrt[3]{3\sqrt{561} - 71},$$

i

$$x_3 = \frac{1}{3} + \frac{1 - i\sqrt{3}}{3\sqrt[3]{3\sqrt{561} - 71}} - \frac{1}{6}\left(1 + i\sqrt{3}\right)\sqrt[3]{3\sqrt{561} - 71}.$$

U ovakvim i sličnim problemima se pribegava numeričkim metodama.

Kroz ovaj rad ćemo predstaviti osnovne iterativne metode za rešavanje nelinearnih jednačina i sistema. Naravno, pored matematičke osnove, algoritme ćemo implementirati u programskom paketu `Mathematica` i na konkretnim primerima videti značaj numeričke matematike. Naravno bez računara, ovi algoritmi bi bili jako zamorni za ručno izvršavanje a nekad i nemogući.

1.1 Šta je to programski paket `Mathematica` i čemu služi?

`Mathematica` je programski paket za matematičke i uopšte naučne primene. `Mathematica` ima velike mogućnosti. Posebno je pogodna za:
1. Numerička izračunavanja,
2. Simbolička izračunavanja,
3. Grafičko prikazivanje podataka i funkcija.

[1]Gerolamo Cardano (1501-1576)-italijanski matematičar

Razlikuje se od klasičnih proceduralnih programskih jezika kao što su `Pascal`, `Fortran` ili `C` po tome što pomenuti programski jezici imaju oko 30 ugrađenih matematičkih operacija, dok ih `Mathematica` ima nekoilko hiljada.

Nastanak programskog paketa `Mathematica` obeležava početak modernog naučnog izračunavanja. Početkom 1960-tih godina nastali su paketi koji su bili specijalizovanji za numerička, algebarska i grafička izračunavanja. Značaj programskog paketa `Mathematica` je taj što ih je ona sve ujedinila u jedinstvenu celinu.

Programski paket `Mathematica` je nastao u softverskoj kompaniji Wolfram Research vlasnika Stivena Volframa[2] i predstavlja jednu od najvećih aplikacija ikada razvijenih. Danas je sastavni deo mnogih programa na studijama matematike, računarskim i tehničkim naukama širom sveta.

1.2 Šta je cilj ovog rada?

Cilj ovog rada je da pokaže značaj progrmskog paketa `Mathematica` u Numeričkoj matematici. Naravno, neće biti reči samo o numeričkim izračunavanjima u paketu `Mathematica` nego i o simboličkom računu i njihovim međusobnim vezama. Videće se i direktna veza između matematike kao teorijske nauke i njenih primena u konkretnim problemima uz pomoć računara.

Svaki numerički algoritam biće prezentovan na strogim matematičkim osnovama i biće implementiran u paketu `Mathematica` uz objašnjenja koje su ugrađene funkcije korišćene.

1.3 Ugrađene funkcije `Solve`, `NSolve` i `FindRoot`

Kao kontrola izvršavanja algoritama koristićemo ugrađene funkcije programskog paketa `Mathematica`:
1. Funkcija `Solve` rešava jednačinu ili sistem jednačina analitički, ako je to moguće, tražeći sva rešenja,
2. Funkcija `NSolve` rešava jednačinu ili sistem jendačina numerički, tražeći sva rešenja jednačine ili sistema,
3. Funkcija `FindRoot` rešava jednačinu ili sistem jednačina u okolini date tačke, tražeći jedno ješenje jednačine ili sistema.

Funkciji `FindRoot` se pribegava ako je jednačina previše analitički komplikovana da bi se tražila sva rešenja ili ako je sistem previše komplikovan da bi mu se našla sva rešenja. Tada se zadaje početna tačka i onda je moguće doći do rešenja.

Primer iz uvoda je na primer rešen koristeći ugrađenu funkciju `Solve` primenjenu na datu jednačinu koja je stepena 3 pa je bilo moguće da se reši analitički. Objasnimo na primerima kako se koriste pomenute funkcije.

1. Pozivom `NSolve[x^3-x^2+x+5==0, x]` dobijamo rešenje smešteno u listu

$$\{\{\ x\to\text{-}1.27816\},\{x\to 1.13908\,\text{-}1.61689\ i\},\{x\to 1.13908\,+1.61689\ i\}\}.$$

Kada bismo hteli da dobijemo rešenje na veći broj tačnih cifara, recimo na primer na 10, pozvali bismo funkciju, gde bismo posle nepoznate promenljive po kojoj se rešava numerički jednačina naveli i broj željenih cifara:

```
NSolve[x^3-x^2+x+5==0, x, 10].
```

[2]Stephen Wolfram(1959)-britanski matematičar, fizičar, programer

2. Pozivom `FindRoot[x^3-x^2+x+5==0,{x ,-1}]` dobijamo jedno rešenje u obliku liste {x→-1.278161486}.
Primetimo da smo tražili rešenje u okolini broja -1. Da smo, na primer, funkciju pozvali na sledeći način:

```
FindRoot[x^3-x^2+x+5==0, {x, 1-I}].
```

dobili bismo listu

{x→1.1390815363990743 -1.6168973891059626 i}.

Primetimo da se, u programskom paketu `Mathematica`, imaginarna jedinica i označava velikim slovom I .

3. Kao što je već napomenuto, paket `Mathematica` podržava simbolličko računanje. Ilustrujmo to sledećim primerom.

Rešimo kvadratnu jednačinu $ax^2 + bx + c = 0$ simbolički.

Pozivom funkcije

```
Solve[ax^2+bx+c==0, x]
```

dobijamo

$$\left\{\left\{\mathtt{x} \to \frac{-\sqrt{\mathtt{b}^2 - 4\mathtt{a}\ \mathtt{c}} - \mathtt{b}}{2\mathtt{a}}\right\}, \left\{\mathtt{x} \to \frac{\sqrt{\mathtt{b}^2 - 4\mathtt{a}\ \mathtt{c}} - \mathtt{b}}{2\mathtt{a}}\right\}\right\}.$$

Primetimo da je jedan od osnovnih tipova podataka sa kojim programski paket `Mathematica` operiše **lista**.

Kada bismo hteli da radimo dalje sa dobijenim vrednostima morali bismo da ih prvo izdvojimo iz date liste. To radimo sledećim nizom naredbi:

```
Solve[ax^2+bx+c==0, x];lista=x/.sol;
```

a prvi element liste izdvaja se kao `lista[[1]]` dok se drugi izdvaja kao `lista[[2]]`.

Rešimo sada simbolički sistem jednačina

$$(\mathcal{S}) : \begin{cases} x + ay + 1 = 0 \\ x + y^2 - 6 = 0, \end{cases},$$

gde je $a \in \mathbb{R}$.

Pozivom `Solve[ay+x+1==0 && x+y^2-6==0,{x,y}]` dobijamo rešenje

$$\left\{\left\{\mathtt{x} \to \frac{1}{2}\left(-\mathtt{a}^2 + \sqrt{\mathtt{a}^2 + 28}\mathtt{a} - 2\right), \mathtt{y} \to \frac{1}{2}\left(\mathtt{a} - \sqrt{\mathtt{a}^2 + 28}\right)\right\}, \left\{\mathtt{x} \to \frac{1}{2}\left(-\mathtt{a}^2 - \sqrt{\mathtt{a}^2 + 28}\mathtt{a} - 2\right), \mathtt{y} \to \frac{1}{2}\left(\sqrt{\mathtt{a}^2 + 28} + \mathtt{a}\right)\right\}\right\}.$$

1.4 Ugrađene funkcije za optimizaciju

Programski paket `Mathematica` raspolaže i optimizacionim funkcijama `Minimize`, `Maximize`, `NMinimize`, `NMaximize`, `FindMinimum` i `FindMaximum`. Treća i četvrta se koriste za numeričku optimizaciju, kada tačne metode ne mogu naći rešenje. Na primer, ako želimo naći minimum funkcije $f(x, y) = x^2 + (xy - 2)^2$ u skupu $\mathbb{R}^2$ pozvali bismo

```
NMinimize [x^2+(xy-2)^2,{x, y}]
```

i dobijamo

$$\{3.68983 \cdot 10^{-6}, \{\mathtt{x} \to 0.00192087, \mathtt{y} \to 1041.19\}\}.$$

Prvi element dobijene liste je traženi minimum a drugi element liste je lista u kojoj su smeštene koordinate dobijenog minimuma.

Funkcije `FindMinimum` i `FindMaximum` nalaze **lokalni ekstremum funkcije** polazeći od zadate tačke dok funkcije `Minimize`, `Maximize`, `NMinimize` i `NMaximize` nalaze **globalni ekstremum funkcije**.

Na primer, pozivom naredbe `FindMinimum[x Cos[x], {x, 2}]` dobijamo lokalni minimum funkcije $f(x) = x\cos x$ u okolini tačke $x = 2$.

Možemo izvršiti minimizaciju neke funkcije sa datim ograničenjima. Na primer, ako se traži minimum funkcije $f(x, y) = x^2 - (y-1)^2$ sa ograničenjem $x^2 + y^2 \leq 4$ pozvali bismo

$$\texttt{NMinimize}\left[\{\texttt{x\^{}2-(y-1)\^{}2}, \texttt{x\^{}2+y\^{}2} \leq 4\}, \{\texttt{x}, \texttt{y}\}\right]$$

i dobijamo rešenje

$$\{-9., \{\texttt{x} \to 6.55605118140317 \cdot 10^{-12}, \texttt{y} \to -2.\}\}.$$

Preciznost izračunavanja kod svih ovih funkcija možemo podešavati parametrom `WorkingPrecision`. Na prethodnom primeru, sa tačnošću na 20 decimalnih mesta, lokalni minimum dobijamo pozivom

$$\texttt{FindMinimum[x Cos[x], \{x, 2\}, WorkingPrecision} -> \texttt{20]}.$$

2 Numerički algoritmi za rešavanje nelinearnih jednačina

U ovom delu razmatraćemo problem određivanja rešenja jednačine $f(x) = 0$. Metode ćemo razvrstavati po brzini nalaženja rešenja i broju računskih operacija koji izvršavaju po iteraciji. Daćemo za svaki algoritam dokaz i implementaciju u programskom paketu `Mathematica`.

Na kraju izdvajamo metode za simultano određivanje nula polinoma. Razlog zašto polinome izdvajamo je taj zato što polinomi imaju mnoge lepih osobina koje druge funkcije ne poseduju:

1) lako se nalazi zbir, proizvod i količnik,

2) lako se računa vrednost polinoma,

3) lako se računaju granične vrednosti, izvodi i integrali.

Zbog ovh lepih osobina polinoma uglavnom se u matematici funkcije aproksimiraju polinomskom funkcijom. Sa tim u vezi dajemo sledeću fundamentalnu teoremu. Dokaz se može naći, na primer, u [1].

Teorema 1. ***(Vajerštras**[3]**)*** *Neka je data neprekidna funkcija $f : [a, b] \to \mathbb{R}$. Tada za svako $\varepsilon > 0$ postoji niz polinoma $(p_n)_{n=1}^{+\infty} \in \mathbb{R}[x]$ koji uniformno konvergira ka f.*

Napomenimo samo još da se kod aproksimacije funkcija u praksi ne koristi Vajerštrasova teorema. Razlog za to je zato što niz iz tvrđenja teoreme vrlo sporo konvergira ka funkciji koju aproksimiramo. Postoje primeri kada za neku jednostavnu funkciju polinom koji se dobija iz Vajerštrasove teoreme mora da bude stepena većeg od 100 što je za dalja korišćenja nefunkcionalno. Dakle, Vajerštrasova teorema ima veliki teorijski značaj, dok nije od praktičnog značaja.

[3]Karl Theodor Wilhelm Weierstrass (1815-1897)-veliki nemački matematičar

2.1 Iterativni procesi i red konvergencije iterativnog metoda

Sada ćemo opisati kako je na izvestan način moguće izmeriti brzinu konvergencije nekog niza.

Definicija 1. *Neka je dat realni niz tačaka* $(x_n)_{n=0}^{+\infty}$ *koji konvergira ka* $\overline{x} \in \mathbb{R}$. *Niz* $(x_n)_{n=0}^{+\infty}$ *ima* ***red konvergencije*** *najmanje* r *ako važi*

$$|x_{n+1} - \overline{x}| = \mathcal{O}(|x_n - \overline{x}|^r).$$

Niz $(x_n)_{n=0}^{+\infty}$ *ima red konvergencije tačno* r *ako važi*

$$\lim_{n \to +\infty} \frac{|x_{n+1} - \overline{x}|}{|x_n - \overline{x}|^r} = C.$$

Konstanta C *se naziva* ***faktor konvergencije*** *ili* ***asimptotska konstanta greške***.

Ako je $r = 1$ reč je o **linearnoj**, za $r = 2$ o **kvadratnoj** a za $r = 3$ o **kubnoj** konvergenciji. Primetimo da kod metoda koji imaju isti red konvergencije, možemo reći da je brži onaj kome je manja asimptotska konstanta greške. Naredna teorema daje način za izračunavanje reda konvergencije određenih iterativnih metoda.

Teorema 2. *Neka iterativni metod* $x_{n+1} = g(x_n)$ *konvergira ka* $\overline{x} \in (a, b)$ *na segementu* $[a, b]$ *i neka funkcija* g *ima neprekidni izvod reda* r *u nekoj okolini tačke* $\overline{x}$. *Tada je red konvergencije iterativnog metoda* r *ako i samo ako važe sledeći uslovi:*

1) $g(\overline{x}) = \overline{x}$,
2) $g'(\overline{x}) = g''(\overline{x}) = ... = g^{(r-1)}(\overline{x}) = 0$,
3) $g^{(r)}(\overline{x}) \neq 0$.

Osim toga važi i

$$\lim_{n \to +\infty} \frac{|x_{n+1} - \overline{x}|}{|x_n - \overline{x}|^r} = \frac{g^{(r)}(\overline{x})}{r!}.$$

Dokaz. Neka su x_{n+1} i x_n dve susedne iteracije koje pripadaju okolini $U(\overline{x})$ u kojoj je $g^{(r)}$ neprekidna funkcija. Na osnovu Tejlorove[4] formule za funkciju g u okolini tačke $\overline{x}$ (sa ostatkom u Košijevom obliku) imamo:

$$g(x_n) = g(\overline{x}) + g'(\overline{x})(x_n - \overline{x}) + \frac{g''(\overline{x})}{2!}(x_n - \overline{x})^2 + ... + \frac{g^{(r-1)}(\overline{x})}{(r-1)!} + \frac{g^{(r)}(\xi_n)}{r!}(x_n - \overline{x})^r,$$

gde je ξ_n tačka koja se nalazi između tačaka x_n i $\overline{x}$. Odavde, na osnovu neprekidnosti funkcije $g^{(r)}$ u $U(\overline{x})$ i toga što se ξ_n nalazi između x_n i $\overline{x}$ imamo koristeći uslove teoreme:

$$\lim_{n \to +\infty} \frac{|x_{n+1} - \overline{x}|}{|x_n - \overline{x}|^r} = \left| \frac{g^{(r)}(\xi_n)}{r!} \right| = \left| \frac{g^{(r)}(\overline{x})}{r!} \right|.$$

Odavde imamo direktno po definiciji da je red konvergencije ovog iterativnog metoda jednak r.

□

[4]Brook Taylor (1685-1731)-engleski matematičar

Primetimo da na osnovu prethodne teoreme sledi da je red konvergencije iterativnih metoda tipa $x_{n+1} = g(x_n)$ uvek prirodan broj i da je faktor konvergencije jednak

$$C = \left| \frac{g^{(r)}(\overline{x})}{r!} \right|.$$

Primetimo takođe da sa povećanjem reda konergencije raste i brzina metoda, ali povećavanjem reda raste i broj funkcijskih izračunavanja po iteraciji. Postoje metodi koji imaju manji red od nekih drugih metoda ali su efiksaniji. Posledica ovog je sledeća definicija.

Definicija 2. *Neka je dat iterativni metod reda r i neka se u svakoj iteraciji zahteva ukupno θ funkcijskih izračunavanja (vreednosti funkcije ili njenih izvoda). Uvedimo sledeće veličine:*

$$E_i = \frac{r}{\theta}, \quad E_c = r^{\frac{1}{\theta}}.$$

*Prva veličina se naziva **informaciona efikasnost** a druga **računska efikasnost**.*

Za sve metode koji slede izračunaćemo red konvegrancije, asimptotsku konstantu greške, informacionu i računsku efikasnost i uporediti ih po tome. Prirodno se nameće pitanje nalaženja **optimalnog** odnosa između reda konvergencije metoda i funkcijskih izračunavanja po iteraciji. Sledeću hipotezu su postavili Kung i Traub 1974. godine i još uvek nije ni dokazana ni opovrgnuta. Na osnovu nje ćemo dati definiciju optimalnog metoda.

Definicija 3. *Iterativni metod je metod **bez memorije** ako je za sračunavanje svake naredne iteracije je potrebna samo vrednost prethodno sračunate iteracije. U protivnom kažemo da je iterativni metod **sa memorijom**.*

Hipoteza 1. ***(Kung, Traub, 1974.)***
Iterativni metod bez memorije koji u svakoj iteraciji zahteva ukupno $m+1$ funkcijskih izračunavanja, ima red konvergencije najviše 2^m.

Definicija 4. *Iterativni metod bez memorije je **optimalan** ako zadovoljava granicu koju predviđa Kung-Traubova hipoteza.*

2.2 Izlazni kriterijumi iterativnih metoda

Da bismo primenili neki iterativni metod, potrebno je definisati način njegovog zaustavljanja, jer nije moguće uraditi beskonačno mnogo iteracija. Dakle, moramo na neki način utvrditi kako ćemo meriti koliko smo se približili tačnom rešenju jednačine.

Dakle, neka je dat konvergentan iterativni metod $x_{k+1} = \Phi(x_k)$ koji konvergira ka $\overline{x}$, tačnom rešenju jednačine $f(x) = 0$.

U ovom radu ćemo koristi kriterijum **uzastopnih apsolutnih razlika** tj. $|x_{k+1} - x_k| < \varepsilon$. U opštem slučaju ovaj kriterijum ne garantuje da smo našli rešenje jednačine sa tačnošću $\varepsilon > 0$.

Još jedan od mogućih kriterijuma je $|f(x_n)| < \varepsilon$. Nedostatak ovog kriterijuma je taj što on može biti zadovoljen i kada je x_n dosta daleko od $\overline{x}$. To se često događa ako je $f'(\overline{x})$ malo.

Moguće je koristiti kriterijum **apsolutne relativne razlike** $|x_{k+1}-x_k|/|x_k| < \varepsilon$. Ovaj kriterijum ne treba koristiti ako je $\overline{x}$ blisko nuli.

Najčešće se zadaje i **maksimalni broj iteracija**. Ako se ovaj broj prekorači, izračunavanja se prekidaju i ispisuje se poruka da rešenje nije pronađeno sa željenom tačnošću.

2.3 Opšti iterativni metod

Sada ćemo preći na konstrukciju konkretnog iterativnog metoda. Pre toga navodimo jednu od fundamentalnih teorema matematike. Teoremu dokazujemo, jer je dokaz bitan za zaustavni kriterijum iterativnog metoda koji sledi. Teorema će biti data u terminima metričkih prostora. Inače, teoremu je dao Banah[5] u svom doktorskom radu 1922. godine.

Podsetimo, preslikavanje f metričkog prostora $(\mathcal{X}, d)$ je kontrakcija ako postoji realan broj $q \in (0,1)$ tako da za sve $x, y \in \mathcal{X}$ važi $d(f(x), f(y)) \leq q \cdot d(x, y)$.

Teorema 3. ***(Banah)*** *Neka je dat kompletan metrički prostor $(\mathcal{X}, d)$ i kontraktivno preslikavanje $f : \mathcal{X} \to \mathcal{X}$. Tada preslikavanje f ima jednistvenu fiksnu tačku u $\mathcal{X}$.*

Dokaz. Neka je $x_0 \in \mathcal{X}$ proizvoljna tačka. Generišimo niz $(x_n)_{n=0}^{+\infty}$ sa $x_{n+1} = f(x_n)$. Dokažimo najpre da je ovako definisan niz Košijev u $\mathcal{X}$. Najpre primetimo da važi sledeća procena za svako $n \in \mathbb{N} \cup \{0\}$ i za neko $q \in (0,1)$:

$$d(x_{n+1}, x_n) = d(f(x_n), f(x_{n-1})) \leq q d(x_n, x_{n-1}).$$

Neka je $m > n$. Na osnovu nejednakosti trougla imamo:

$$\begin{aligned} d(x_m, x_n) &\leq d(x_m, x_{m-1}) + ... + d(x_{n+1}, x_n) \\ &\leq (q^{m-1} + ... + q^n) d(x_1, x_0) \leq \frac{q^n}{1-q} d(x_1, x_0). \end{aligned}$$

Odavde iz toga što je $q < 1$ kada $m \to +\infty$ dobijamo da $d(x_m, x_n) \to 0$, odnosno da je niz $(x_n)_{n=0}^{+\infty}$ Košijev u $\mathcal{X}$.

Kako je prostor $(\mathcal{X}, d)$ po uslovu teoreme kompletan imamo da je niz $(x_n)_{n=0}^{+\infty}$ konvergentan u $\mathcal{X}$. Neka je

$$\lim_{n \to +\infty} x_n = \overline{x} \in \mathcal{X}.$$

Dokažimo da je $\overline{x}$ fiksna tačka preslikavanja f. Na osnovu procene:

$$\begin{aligned} d(\overline{x}, f(\overline{x})) &\leq d(\overline{x}, x_n) + d(x_n, f(x_n)) + d(f(x_n), f(\overline{x})) \\ &\leq d(\overline{x}, x_n) + d(x_n, x_{n+1}) + q d(x_n, \overline{x}) \leq (1+q) d(\overline{x}, x_n) + q^n d(x_1, x_0). \end{aligned}$$

Kako desna strana nejednakosti teži nuli kada $n \to +\infty$, dobijamo da je

$$d(\overline{x}, f(\overline{x})) = 0$$

odakle sledi $\overline{x} = f(\overline{x})$.
Dokažimo da je fiksna tačka jedinstvena. Ako bi postojale dve različite fiksne tačke $\overline{x_1}$ i $\overline{x_2}$, tada bi bilo

$$d(\overline{x_1}, \overline{x_2}) = d(f(\overline{x_1}), f(\overline{x_2})) \leq q d(\overline{x_1}, \overline{x_2}),$$

što je nemoguće, jer je $q \in (0,1)$. □

[5]Stefan Banach (1892-1945)-poljski matematičar

Najvažnija nejednakost za numeričku matematiku i primene, koja je posledica dokaza ove teoreme, je sledeća:

Kako za sve $m, n \in \mathbb{N}$ takve da $m > n$ važi

$$d(x_m, x_n) \leq \frac{q^n}{1-q} d(x_1, x_0),$$

a preslikavanje d kao metrika na $\mathcal{X}$ je neprekidno na $\mathcal{X}$, imamo kada $m \to +\infty$ u poslednjoj nejednakosti da za svako $n \in \mathbb{N}$ važi

$$d(\overline{x}, x_n) \leq \frac{q^n}{1-q} d(x_1, x_0).$$

Ova nejednakost nam omogućuje da direktno procenimo grešku i izračuanamo koliko je iteracija potrebno da bi se zadovoljila željena tačnost. Sve ostalo je posao za računar. Preciznije, da bi se zadovoljila tačnost za neko $\varepsilon > 0$ potrebno je da važi:

$$d(\overline{x}, x_n) \leq \frac{q^n}{1-q} d(x_1, x_0) < \varepsilon \Longleftrightarrow n > \frac{\frac{\ln(1-q)\varepsilon}{d(x_0,x_1)}}{\ln q}.$$

Dakle, ako sračunamo $1 + \left[\frac{\frac{\ln(1-q)\varepsilon}{d(x_0,x_1)}}{\ln q} \right]$ iteracija, sigurni smo da smo zadovoljili željenu tačnost. U praksi je situacija još povoljnija, jer smo imali nekoliko majorizacija tako da se uglavnom dobija tačnost za red veličine više. Ovu pogodnost ćemo videti na konkretnim numeričkim primerima koji slede.

2.3.1 Algoritam

Teorema 4. *Neka je data funkcija $\Phi : [a, b] \to [a, b]$ koja je neprekidna na $[a, b]$ i diferencijabilna u (a, b). Ako postoji $q \in (0, 1)$ tako da je $|\Phi'(x)| \leq q < 1$ za svako $x \in (a, b)$, onda je iterativni metod $x_{n+1} = \Phi(x_n)$ konvergentan za svako $x_0 \in [a, b]$, a granična vrednost $\overline{x}$ predstavlja jedinstveno rešenje jednačine $\Phi(x) = x$ na segmentu $[a, b]$.*

Dokaz. Neka su $x, y \in [a, b]$. Primenom Lagranžove teoreme na funkciju Φ dobijamo

$$|\Phi(x) - \Phi(y)| = |\Phi'(\xi)||x - y|,$$

gde je ξ tačka između x i y. Odavde, koristeći $|\Phi'(x)| \leq q < 1$, sledi da je Φ kontrakcija, a kako je prostor $(\mathbb{R}, |\cdot|)$ kompletan metrički prostor, imamo na osnovu Banahove teoreme naše tvrđenje. □

Iterativni postupak prekidamo kada sračunamo $1 + \left[\frac{\ln(1-q)\varepsilon}{|x_0 - \Phi(x_1)| \ln q} \right]$ iteracija, i ovaj kriterijum garantuje željenu tačnost.

Alternativni kriterijum, koji ne garantuje željenu tačnost, ali u praksi se pokazuje kao vrlo uspešan je taj da izračunavanja prekinemo kada bude zadovoljeno $|x_{n+1} - x_n| < \varepsilon$.

Formalno, radi preglednosti, dajemo algoritam u obliku **pseudokod**[6] notacije.

[6]Pseudokod notacija liči na strukturu nekog proceduralnog programskog jezika, ali se mnogi čisto implementacioni detalji ne prikazuju

Algoritam 1 Opšti iterativni metod

Input Iterativna funkcija Φ, segment $[a, b]$, početna aproksimacija $x_0 \in [a, b]$ i tačnost ε.

1: $n := 1$
2: $x_1 := \Phi(x_0)$;
3: **while** $|x_n - x_{n-1}| \geq \varepsilon$ **do**
3: pom $:= x_n$
4: $x_n := \Phi(\text{pom})$
5: $x_{n-1} :=$ pom
6: $n := n + 1$
8: **end while**
7: **return** x_n

2.3.2 Implementacija

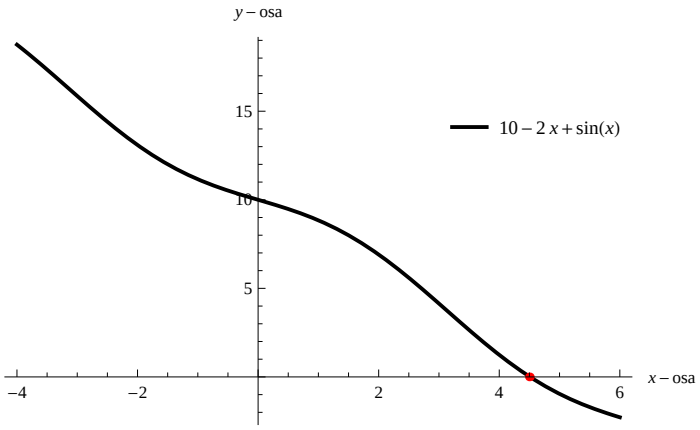
Sada dajemo implementaciju opšteg iterativnog metoda u programskom paketu `Mathematica`. Kod će biti dat u formi **modula** da bi se lakše menjali ulazni parametri. Ulazni parametri ce biti iterativna funkcija `Phi`, željena tačnost `eps` i početna aproksimacija `x0`. Naravno, korisnik je obavezan da proveri pre toga da li su zadovoljeni uslovi teoreme da bi algoritam korektno radio.

```
OpstiIterativni[Phi_, eps_, x0_] :=
 Module[{xnew, xold, i, Tablica = {{"k", "x[k]"}}, pom},
  i = 0;
  xold = x0;
  AppendTo[Tablica, {i, N[xold, 12]}];
  xnew = Phi[xold];
  i = 1;
  While[Abs[xold - xnew] >= eps,
   AppendTo[Tablica, {i, N[xnew, 12]}];
   pom = xnew;
   xnew = Phi[pom];
   xold = pom;
   i++];
  Return[TableForm[Tablica]]
  ]
```

Primer 1. *Rešiti numerički jednačinu* $10 - 2x + \sin x = 0$ *pomoću metoda proste iteracije, ako je moguće, sa tačnošću* $\varepsilon = 10^{-8}$.

Rešenje. Neka je $f(x) = 10 - 2x + \sin x,\ x \in \mathbb{R}$. Kako je $f'(x) = -2 + \cos x < 0$ imamo da f opada na celom skupu $\mathbb{R}$. Dakle, ako funkcija ima nulu onda je ona (zbog monotonosti) jedinstvena. Lokalizaciju možemo utvrditi na sledeća dva načina. Na primer, možemo primetiti da je $f(4) > 0$ a $f(5) < 0$ pa kako je funkcija neprekidna (videti teoremu o međuvrednosti iz narednog odeljka) zaključujemo da ima nulu na $[4, 5]$.

Drugi način je da nacrtamo grafik ove funkcije. U tu svrhu, u programskom paketu `Mathematica` postoji ugrađena funkcija `Plot` koja crta grafike funkcija. Primera radi, pozivom funkcije `Plot[10-2x+Sin[x],{x,-4,5}]` dobijamo grafik funkcije na segmentu $[-4, 5]$.



Dakle, i sa grafika možemo zaključiti da jednačina ima nulu na $[4, 5]$. Napišimo jednačinu u njenom ekvivalentnom obliku

$$x = 5 + \frac{1}{2}\sin x.$$

Dokažimo da je iterativna funkcija $\Phi : [4, 5] \to [4, 5]$, definisana sa $\Phi(x) = 5 + \frac{1}{2}\sin x$, kontrakcija. Kako je $|\Phi'(x)| = |0.5\cos x| \leq 0.5$, imamo kontraktivnost funkcije Φ. Dakle, možemo primeniti opšti iterativni metod. Neka je

$$x_{k+1} = 5 + 0.5\sin x_k, \quad k \in \mathbb{N} \cup \{0\}.$$

Sada dobijamo rešenje pozivajući `OpstiIterativni`[Phi, 10^(-8), 4.5]. Rešenje je prikazano, radi preglednosti, tabelarno:

k	x[k]
0	**4.5**0000000000000
1	**4.51**123494116745
2	**4.51**008167346816
3	**4.510**19721082255
4	**4.5101**8560662482
5	**4.51018**677181749
6	**4.510186**65481599
7	**4.5101866**6656453

Primetimo da smo rešenje dobili sa željenom tačnošću, na osam tačnih cifara. Tačne cifre su boldovane. Analizirajući vrednosti dobijene u tabeli, možemo primetiti tipičnu linearnu konvergenciju metoda. U svakoj sledećoj iteraciji je dobijena još po jedna tačna cifra. △

2.4 Metod polovljenja intervala

U ovom delu ćemo opisati jedan od najprostijih iterativnih metoda. Bazira se na istom principu kao i **binarna pretraga**[7] ali je jako spor dok je binarna pretraga

[7]Binarna pretraga je algoritam za pronalaženje elementa u sortiranom nizu brojeva. Neka je niz celih brojeva sortiran u neopadajućem poretku. Odabere se središnji element niza, ako je trazeni broj veći od odabranog korak se ponavlja na levoj polovini niza a ako je manji na desnoj. Ako niz ima n elemenata složenost ovog algoritma je $\mathcal{O}(\log n)$. Za više informacija videti izuzetnu knjigu [11].

jedan od sofisticiranijih načina za nalaženje nekog elementa u sortiranom nizu. Iako je dosta spor (prvog reda), primenljiv je na dosta široku klasu funkcija. Naime, zahteva se samo neprekidnost funkcije na segmentu gde je izolovano jedno rešenje jednačine.

2.4.1 Algoritam

Iskažimo najpre dobro poznatu teoremu koja će nam pomoći oko konstrukcije algoritma. Njen dokaz se može naći na primer u [8].

Teorema 5. ***(O međuvrednosti)*** *Neka je funkcija $f : [a, b] \to \mathbb{R}$ neprekidna i neka na krajevima segmenta uzima vrednosti različitog znaka. Tada postoji $\xi \in (a, b)$ tako da je $f(\xi) = 0$.*

Sada možemo konstruisati algoritam. Odaberemo središnju tačku $\frac{a+b}{2}$. Ako je $f(\frac{a+b}{2}) = 0$ onda je nađena nula funkcije.

Ako nije primenjujemo prethodni korak na jedan od segmenata $[a, \frac{a+b}{2}]$ ili $[\frac{a+b}{2}, b]$ u zavisnosti od toga na kom segmentu funkcija ima različit znak u krajevima.

Dokaz da metod konvergira ka izolovanom rešenju jednačine na segmentu je direktna posledica sledeće teoreme.

Teorema 6. ***(Kantor[8])*** *Neka je $[a_n, b_n]$ niz **umetnutih** segmenata čija dužina teži 0. Tada je presek svih segmenata iz datog niza jednoelementan skup.*

Dokaz. Posmatrajmo nizove a_n i b_n, levih i desnih krajeva svakog od umetnutih segmenata. Kako je $(a_n)_{n=0}^{+\infty}$ rastući i odozgo ograničen sa b_0, imamo da je on konvergentan. Neka je

$$L = \lim_{n\to\infty} a_n \in \mathbb{R}.$$

Isto tako, posmatrajući niz $(b_n)_{n=0}^{+\infty}$ zaključujemo da je on opadajući i ograničen odozdo sa a_0, te je i on konvergentan. Neka je

$$D = \lim_{n\to\infty} B_n \in \mathbb{R}.$$

Po uslovu teoreme, na osnovu rečenog, dobijamo da važi:

$$a_0 \leq a_n \leq L \leq D \leq b_n \leq b_n, \quad \forall n \in \mathbb{N}.$$

Kako dužina segmenata teži 0 kada se n neograničeno povećava, prelaskom u poslednjoj nejednakosti na granični proces kada $n \to +\infty$ dobijamo, koristeći "teoremu o dva policajca", da je $L = D = \xi$. Dokažimo da je ξ zajednička tačka svim segmentima.

Kako je

$$\xi \in [a_n, b_n], \quad \forall n \in \mathbb{N}$$

onda važi i

$$\{\xi\} \in \bigcap_{n=0}^{+\infty} [a_n, b_n].$$

[8]Georg Ferdinand Ludwig Philipp Cantor (1845-1918)-veliki nemački matematičar

Dakle, pokazali smo da je

$$\{\xi\} \subseteq \bigcap_{n=0}^{+\infty} [a_n, b_n].$$

Dokažimo obrnutu inkluziju. Ako bi postojalo neko ξ_1 tako da je $\xi \neq \xi_1$ i

$$\xi_1 \in \bigcap_{n=0}^{+\infty} [a_n, b_n],$$

tada bi bilo

$$L \leq \xi_1 \leq D$$

i

$$L \leq \xi \leq D,$$

pa kako dužina segmenata teži 0 imali bismo da je $\xi = \xi_1$. Dakle, važi

$$\{\xi\} = \bigcap_{n=0}^{+\infty} [a_n, b_n].$$

□

Posledica 1. *(**O konvergenciji metoda polovljenja intervala**) Neka je $f : [a, b] \to \mathbb{R}$ neprekidna funkcija i neka na krajevima segmenta uzima vrednosti različitog znaka. Tada je metod intervala konvergentan. Presek svih segmenata je rešenje jednačine $f(x) = 0$.*

Dokaz. Neka je $[a_n, b_n]$ niz umetnutih segmenata dobijenih metodom polovljenja intervala. Kako dužina segmenata teži nuli, po Kantorovoj teoremi imamo da je presek svih segmenata jednoelementan skup. Neka je

$$\bigcap_{n=0}^{+\infty} [a_n, b_n] = \xi,$$

Dokažimo da je $f(\xi) = 0$. Neka je

$$L = \lim_{n\to\infty} a_n = \xi = \lim_{n\to\infty} b_n = D.$$

Koristeći neprekidnost funkcije f, imamo da je

$$\lim_{n\to\infty} f(a_n)f(b_n) = f\left(\lim_{n\to\infty} a_n\right) f\left(\lim_{n\to\infty} b_n\right) = f(L)f(D) = f(\xi)^2 \leq 0.$$

Poslednje može da važi samo kada je $f(\xi) = 0$. Time smo dokazali da je metod polovljenja intervala konvergentan.

□

Izračunavanja vršimo sve dok je dužina tekućeg intervala veća ili jednaka od željene tačnosti ε. U praksi se ne ispituje uslov da li je vrednost središnje tačke 0, jer radimo sa numeričkim vrednostima pa će uvek biti vrednost središnje tačke različita od 0.

Teorijski, moguće je izračunati koliko nam je potrebno koraka do postizanja željene tačnosti ε. Naime, kako mi u svakom sledećem koraku interval podelimo na

Algoritam 2 Metod polovljenja intervala

Input Funkcija f, segment $[a_0, b_0]$ takav da je $f(a_0)f(b_0) < 0$ i tačnost ε.

$n := 0$;
while $|b_n - a_n| \geq \varepsilon$ **do**
  $s_n := (a_n + b_n)/2$
  if $f(s_n) = 0$ **then**
    return s_m
  else if $f(a_n)f(b_n) < 0$ **then**
    $a_{n+1} := a_n,\ b_{n+1} := s_n$
  else
    $b_{n+1} := b_n,\ a_{n+1} := s_n$
  end if
$n := n + 1$
end while
return s_n

pola imamo da će u n-tom koraku dužina intervala biti $\frac{b-a}{2^n}$. Greška aproksimacije je

$$|\overline{x} - s_n| \leq \frac{b_n - a_n}{2} = \frac{b-a}{2^{n+1}},$$

gde je $s_n = (a_n + b_n)/2$. Dakle, potreban broj iteracija određujemo iz uslova

$$\frac{b-a}{2^{n+1}} < \varepsilon \Longleftrightarrow n > \log_2 \frac{b-a}{\varepsilon} - 1.$$

Zato, ukoliko sračunamo

$$\left[\log_2 \frac{b-a}{\varepsilon} - 1\right] + 1$$

iteracija sigurni smo da smo zadovoljili željenu tačnost ε.

Algoritam 3 Metod polovljenja intervala sa unapred određenim brojem iteracija

Input Funkcija f, segment $[a_0, b_0]$ takav da je $f(a_0)f(b_0) < 0$ i tačnost ε.

$n := 0$;
while $n \leq \left[\log_2 \frac{b-a}{\varepsilon} - 1\right] + 1$ **do**
  $s_n := (a_n + b_n)/2$
  if $f(s_n) = 0$ **then**
    return s_m
  else if $f(a_n)f(b_n) < 0$ **then**
    $a_{n+1} := a_n,\ b_{n+1} := s_n$
  else
    $b_{n+1} := b_n,\ a_{n+1} := s_n$
  end if
$n := n + 1$
end while
return s_n

Pseudokod sa unapred određenim brojem iteracija bi bio, prema prethodno rečenom:

Spora konvergentnost metoda je nadoknađena mogućnošću njegove širkoke upotrebe, jer se samo od uslova zahteva neprekidnost na segmentu i da funkcija uzima vrednosti različitih znakova na krajevima tog segmenta.

Takođe nema opasnosti od deljenja brojevima bliskim nuli, kao što će se videti da postoji kod drugih (bržih) metoda. Dakle, ovaj metod poseduje svojstvo **globalne stabilnosti**.

Primetimo da je broj koraka do postizanja željene tačnosti logaritamski u odnosu na dužinu intervala, što predstavlja analogiju sa već pomenutom binarnom pretragom. Ova metoda nema nekog praktičnog značaja dok je teorijski značaj fundamentalan.

2.4.2 Implementacija

Na osnovu pseudokod notacije lako sprovodimo implementaciju algoritma metoda polovljenja intervala u bilo kom programskom jeziku. Kod dajemo u programskom paketu `Mathematica`.

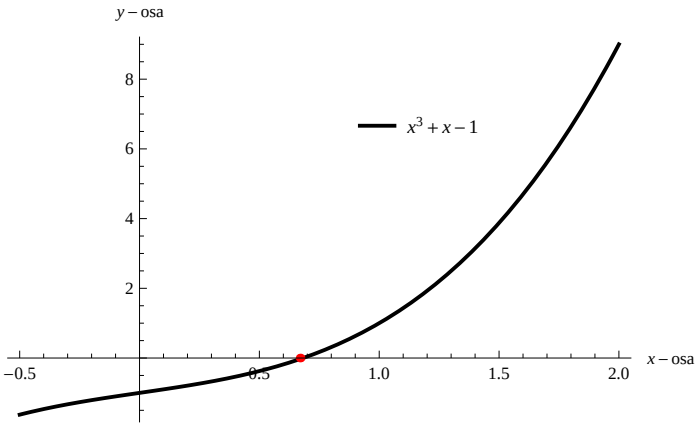
```
MetodPolovljenja[f_, a_, b_, eps_] :=
 Module[{Tablica = {{"k", "x[k]"}}, i, sred, A, B},
  i = 0;
  A = a;
  B = b;
  While[ Abs[B - A] > eps,
   sred = (A + B)/2;
   AppendTo[Tablica, {i, N[sred, 12]}];
   If[f[sred] == 0, Print[sred]];
   If[f[sred]*f[A] < 0, B = sred, A = sred];
   i++
   ];
  Return[TableForm[Tablica]]
  ]
```

Primer 2. *Rešiti numerički, metodom polovljenja jednačinu* $x^3 + x - 1 = 0$ *sa tačnošću* $\varepsilon = 10^{-5}$.

Rešenje. Neka je funkcija $f : \mathbb{R} \to \mathbb{R}$ definisana sa

$$f(x) = x^3 + x - 1.$$

Kako je $f'(x) = 3x^2 + 1 > 0$ imamo da je funkcija f rastuća na $\mathbb{R}$. Sračunavajući $f(0) = -1 < 0$ i $f(1) = 1 > 0$ zaključujemo po teoremi o međuvrednosti funkcije da data jednačina ima jedinstvenu nulu na segmentu $[0, 1]$.



k	x[k]	k	x[k]
1	**0**.500000000000	1	**0**.500000000000
2	**0**.750000000000	2	**0**.750000000000
3	**0**.625000000000	3	**0**.625000000000
4	**0.6**87500000000	4	**0.6**87500000000
5	**0.6**56250000000	5	**0.6**56250000000
6	**0.6**71875000000	6	**0.6**71875000000
7	**0.6**79687500000	7	**0.6**79687500000
8	**0.6**83593750000	8	**0.6**83593750000
9	**0.68**1640625000	9	**0.68**1640625000
10	**0.68**2617187500	10	**0.68**2617187500
11	**0.682**128906250	11	**0.682**128906250
12	**0.682**373046875	12	**0.682**373046875
13	**0.682**250976562	13	**0.682**250976562
14	**0.682**312011719	14	**0.682**312011719
15	**0.6823**42529297	15	**0.6823**42529297
16	**0.6823**27270508	16	**0.6823**27270508
17	**0.6823**34899902	17	**0.6823**34899902

U prvoj tabeli su rezultati dobijeni korišćenjem izlaznog kriterijuma $|x_n - x_{n-1}| < \varepsilon$ dok su u drugoj rezultati dobijeni korišćenjem kriterijuma sa unapred sračunatim brojem iteracija. Primetimo da se u ovom slučaju dobijaju identični rezultati. $\triangle$

2.5 Njutnov metod

U praksi, jedan od najprimenjenijih metoda je Njutnov[9] metod. On sadrži u sebi fundamentalnu ideju o **linearizaciji** problema. U našem konkretnom slučaju rešavamo nelinearnu jednačinu $f(x) = 0$. Kako nemamo opšti metod za njeno rešavanje a znamo opšti metod za rešavanje linearnih jednačina, pokušaćemo da problem svedemo na problem rešavanja linearnih jednačina. Koristićemo se pomenutnom Tejlorovom formulom, razviijajuću funkciju f u okolini tačnog rešenja.

Ovakav pristup problemu ima sledeće pogodnosti:

1. Metod koji dobijemo će biti brži nego metod polovljenja ili opšti iterativni metod, jer koristimo dodatnu informaciju o prvom izvodu funkcije,

[9]Isaac Newton (1643-1727)-veliki engleski matematičar, fizičar i filozof

2. Iterativna formula se ne komplikuje.

Negativne strane ovakvog pristupa su sledeće:

1. Zahteva se diferencijabilnost funkcije pa se smanjuje klasa funkcija na koju možemo metod primenjivati,

2. Oblast konvergencije je uža (**lokalno konvergentan** metod) nego kod metoda polovljenja ili opšteg iterativnog metoda.

Nekad se Njutnov metod naziva **metod tangente** jer je pomenuta linearizacija ništa drugo nego zamenjivanje funkcije tangentnom pravom.

2.5.1 Algoritam

Konstrukcija algoritma, na osnovu izložene ideje, bi bila:

Polazimo od početne aproksimacije x_0. Postavljamo tangentu grafika funkcije $y = f(x)$ u tački $(x_0, f(x_0))$. Narednu aproksimaciju nalazimo u tački gde tangenta seče x-osu.

Formlano neka je x_n n-ta aproksimacija rešenja jednačine $f(x) = 0$. Jednačina tangente grafika funkcije $y = f(x)$ u tački $(x_n, f(x_n))$ je

$$(t_n): \quad y - f(x_n) = f'(x_n)(x - x_n), \quad n \in \mathbb{N} \cup \{0\}.$$

Narednu aproksimaciju rešenja nalazimo iz preseka tangente t_n i x-ose. Rešavajući (linearnu) jednačinu, dobijamo iterativni metod

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n \in \mathbb{N} \cup \{0\}.$$

Sledi teorema koja daje uslove za konvergenciju Njutnovog metoda.

Teorema 7. *Neka je $f \in \mathcal{C}^{(2)}(a, b)$, pri čemu rešenje $\overline{x}$ jednačine $f(x) = 0$ pripada intervalu (a, b) i za svako $x \in (a, b)$ važi $f'(x) \neq 0$. Tada postoji $\varepsilon > 0$ i postoji segment $[\overline{x} - \varepsilon, \overline{x} + \varepsilon]$ takav da je Njutnov metod konvergentan za svako $x_0 \in [\overline{x} - \varepsilon, \overline{x} + \varepsilon]$.*

Dokaz. Iterativna funkcija Njutnovog metoda je

$$\Phi(x) = x - \frac{f(x)}{f'(x)}.$$

Tada je

$$\Phi'(x) = 1 - \frac{((f'(x))^2 - f(x)f''(x)}{(f'(x))^2} = \frac{f(x)f''(x)}{(f'(x))^2}.$$

Kako je $f(\overline{x}) = 0$ imamo da je $\Phi'(\overline{x}) = 0$. Kako je $f \in \mathcal{C}^{(2)}(a, b)$, imamo da je funkcija $\Phi'(x)$ neprekidna u $\overline{x}$ pa po definiciji neprekidnosti postoji $[\overline{x} - \varepsilon, \overline{x} + \varepsilon] \subseteq (a, b)$ takav da je $\Phi(x) \leq q < 1$ za neko $q \in (0, 1)$ i za svako $x \in [\overline{x} - \varepsilon, \overline{x} + \varepsilon]$. Kako je

$$|\Phi(x) - \Phi(\overline{x})| = |\Phi'(\xi)||x - \overline{x}|$$

gde je ξ između x i $\overline{x}$, dobijamo da je

$$|\Phi(x) - \overline{x}| < |x - \overline{x}| \leq \varepsilon$$

tj. imamo da je

$$\Phi([\overline{x} - \varepsilon, \overline{x} + \varepsilon]) \subseteq [\overline{x} - \varepsilon, \overline{x} + \varepsilon].$$

Primenom posledice Banahove teoreme dobijamo naše tvrđenje.

□

Teorema daje samo egzistenciju takve okoline ali ne daje koja je to okolina. Jedan od najtežih problema iz teorije iterativnih metoda je odabir početne aproksimacije. U praksi stvari stoje malo drugačije. Početna tačka se bira, što bi se reklo, **dovoljno blizu** tačnog rešenja. Potom se pokrene program i ukoliko se vidi da dobijene iteracije konvergiraju, to se uzima za rešenje. Naravno, može se desiti da se dobije nešto što uopšte nije blizu rešenja. Uglavnom su to neki veštački konstruisani slučajevi koji se u praksi ne javljaju često. Na kraju ovog dela daćemo neke ovakve primere gde se ispoljava "patologija" iterativnih metoda u praksi.

Sada dajemo teoremu koja određuje red konvergencije Njutnovog metoda, pod izvesnim uslovima.

Teorema 8. *Neka važe uslovi prethodne teoreme. Ukoliko je* $f''(\overline{x}) \neq 0$*, onda je red konvergencije metoda* $r = 2$ *i asimptotska konstanta greške jednaka* $C = \frac{f''(\overline{x})}{2f'(\overline{x})}$.

Dokaz. Neka je $e_n = x_n - \overline{x}$. Na osnovu Tejlorovih razvoja funkcija f i f' u okolini rešenja $\overline{x}$ imamo:

$$f(x_n) = f(\overline{x}) + f'(\overline{x})e_n + \frac{f''(\overline{x})}{2}e_n^2 + o(e_n^2),$$

$$f'(x_n) = f'(\overline{x}) + f''(\overline{x})e_n + o(e_n).$$

Zamenom u formulu kojom je definisan iterativni metod dobijamo

$$e_{n+1} = e_n - \frac{f'(\overline{x})e_n + \frac{f''(\overline{x})}{2}e_n^2 + o(e_n^2)}{f'(\overline{x}) + f''(\overline{x})e_n + o(e_n)} = \frac{\frac{f''(\overline{x})}{2}e_n^2 + o(e_n^2)}{f'(\overline{x}) + f''(\overline{x})e_n + o(e_n)},$$

a odatle

$$\frac{e_{n+1}}{e_n^2} = \frac{\frac{f''(\overline{x})}{2} + \frac{o(e_n^2)}{e_n^2}}{f'(\overline{x}) + f''(\overline{x})e_n + o(e_n)} \longrightarrow \frac{f''(\overline{x})}{2f'(\overline{x})}, \ (n \to +\infty).$$

Ovime je dokazana naša teorema.

□

Sada dajemo pseudokod ovog iterativnog algoritma.

Algoritam 4 Njutnov metod tangenti

Input Funkcija f, segment $[a, b]$ gde je izolovan koren jednačine, početna aproksimacija $x_0 \in [a, b]$ i tačnost ε.

1: $n := 1$
2: $x_1 := x_0 - \frac{f(x_0)}{f'(x_0)}$;
3: **while** $|x_n - x_{n-1}| \geq \varepsilon$ **do**
3: pom $:= x_n$
4: $x_n :=$ pom $- \frac{f(\text{pom})}{f'(\text{pom})}$
5: $x_{n-1} :=$ pom
6: $n := n + 1$
8: **end while**
7: **return** x_n

Može se koristiti sledeća modifikacija Njutnovog metoda, tako što ne računamo vrednost izvoda funkcije u svakoj iteraciji nego samo u prvoj. Preciznije, koristimo sledeću iterativnu formulu

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)}.$$

Ovako imamo manje računa ali se i red konvergencije smanjuje, tj $r = 1$.

Geometrijski gledano, metod funkioniše tako što se tangenta postavlja samo u tački $(x_0, f(x_0))$ a svaka sledeća iteracija se dobija tako što tangentu pomeramo paralelno sa postavljenom i gledamo njene preseke sa $x-$osom. Ovde je i za očekivati da se red konvergencije smanji jer nigde nije suštinski iskorišćena informacija o prvom izvodu funkcije f. Formalno, ovo potvrđuje sledeća teorema.

Teorema 9. *Neka je f neprekidno diferencijabilna funkcija na intervalu (a, b) gde je izolovan koren jednačine $\overline{x}$. Neka za svako $x \in (a, b)$ važi $f'(x) \neq 0$. Tada je red konvergencije Modifikovanog Njutnovog metoda jednak $r = 1$.*

Dokaz. Kako je funkcija f neprekidno diferencijabilna na (a, b) važi

$$f(x_n) = f(\overline{x}) + f'(x_n)(x_n - \overline{x}) + o(x_n - \overline{x}).$$

Kako je $f(\overline{x}) = 0$ dobijamo zamenom u formulu kojom je definisan iterativni metod

$$\frac{x_{n+1} - \overline{x}}{x_n - \overline{x}} = \left(1 - \frac{f'(\overline{x})}{f'(x_0)}\right) + o(x_n - \overline{x}).$$

Prelaskom na granični proces kada $n \to +\infty$ u poslednjoj jednakosti, dobijamo direktno po definiciji, tvrđenje naše teoreme. □

Kod Modifikovanog Njutnovog metoda ne može doći do problema deljenja brojevima bliskim 0, pod uslovom da $f'(x_0)$ nije blisko nuli.

Informaciona efikasnost Njutnovog metoda je $E_i = \frac{2}{2} = 1$ a računska efiksanost je $E_c = 2^{\frac{1}{2}} = \sqrt{2}$.

Kod Njutnovog metoda broj funkcijskih izračunavanja po svakoj iteraciji je 2 a red konvergencije je $r = 2$, To znači da Njutnov metod zadovoljava granicu koju predviđa Kung-Traubova hipoteza pa je optimalan.

Informaciona efikasnost Modifikovanog Njutnovog metoda je $E_i = \frac{1}{1} = 1$ a računska efiksanost $E_c = 1^{\frac{1}{1}} = 1$.

Kod Modifikovanog Njutnovog metoda je broj funkcijskih izračunavanja po svakoj iteraciji jednak 1, ali je red konvergencije takođe jednak 1, te imamo da ovaj metod ne zadovoljava granicu koju predviđa Kung-Traubova hipoteza, te nije optimalan.

Sledi pseudokod Modifikovanog Njutnovog algoritma.

2.5.2 Implementacija

Sada dajemo implementaciju Njutnovog metoda i Modifikovanog Njutnovog metoda u programskom paket `Mathematica`. Na kraju, dajemo poređenje ova dva metoda na konkretnom numeričkom primeru.

Zaustavni kriterijum će biti $|x_n - x_{n-1}| < \varepsilon$ i kod Njutnovog metoda i Modifikovanog Njutnovog metoda, a izračunavanja će biti vršena u double precision aritmetici.

Algoritam 5 Modifikovani Njutnov metod tangenti

Input Funkcija f, segment $[a, b]$ gde je izolovan koren jednačine, početna aproksimacija $x_0 \in [a, b]$ i tačnost ε.

1: $n := 1$
2: $x_1 := x_0 - \frac{f(x_0)}{f'(x_0)}$;
3: **while** $|x_n - x_{n-1}| \geq \varepsilon$ **do**
3: $\quad$ pom $:= x_n$
4: $\quad x_n :=$ pom $- \frac{f(\text{pom})}{f'(x_0)}$
5: $\quad x_{n-1} :=$ pom
6: $n := n + 1$
8: **end while**
7: **return** x_n

```
NjutnovMetod[f_, x0_, eps_] :=
 Module[{Tablica = {{"i", "x[i]"}}, xold, xnew, xpom, i},
  xold = x0;
  i = 0;
  AppendTo[Tablica, {i, N[xold, 12]}];
  i++;
  xnew = xold - f[xold]/f'[xold];
  AppendTo[Tablica, {i, N[xnew, 12]}];
  i++;
  While[Abs[xnew - xold] >= eps,
   xpom = xnew;
   xnew = xpom - f[xpom]/f'[xpom];
   AppendTo[Tablica, {i, N[xnew, 12]}];
   xold = xpom;
   i++
   ];
  Return[TableForm[Tablica]]
  ]


ModifikovaniNjutnovMetod[f_, x0_, eps_] :=
 Module[{Tablica = {{"i", "x[i]"}}, xold, xnew, xpom, i},
  xold = x0;
  i = 0;
  AppendTo[Tablica, {i, N[xold, 12]}];
  i++;
  xnew = xold - f[xold]/f'[x0];
  AppendTo[Tablica, {i, N[xnew, 12]}];
  i++;
  While[Abs[xnew - xold] >= eps,
   xpom = xnew;
   xnew = xpom - f[xpom]/f'[x0];
   AppendTo[Tablica, {i, N[xnew, 12]}];
   xold = xpom;
   i++
```

```
 ];
Return[TableForm[Tablica]]
]
```

Primetimo da smo mogli na početku da zapamtimo vrednost $f'(x_0)$ i nju da koristimo na dalje u toku programa jer na ovaj način opterećujemo izračunavanja. Kod ovog primera ova činjenica ne utiče na brzinu iterativnog procesa, ali kod većih izračunavanja to postaje vrlo osetno.

Primer 3. *Njutnovim i Modifikovanim Njutnovim metodom rešiti jednačinu* $x^3 - 2x - 5 = 0$,[10] *sa tačnošću* $\varepsilon = 10^{-7}$.

Rešenje. Neka je $f(x) = x^3 - 2x - 5$. Kako je $f(2) = -1 < 0$ i $f(3) = 16 > 0$ sledi da postoji barem jedan koren jednačine $f(x) = 0$ na segmentu $[2, 3]$. Sada je iterativni proces definisan sa

$$x_{i+1} = x_i - \frac{x_i^3 - 2x_i - 5}{3x_i^2 - 2}, \quad i \in \mathbb{N} \cup \{0\}.$$

Za početnu aproksimaciju uzimamo $x_0 = 2$. U prvoj tabeli su prikazani rezultati dobijeni pomoću Njutnovog metoda a u drugoj pomoću Modifikovanog Njutnovog metoda. Vidimo da je Njutnov metod znatno brži i postigao je dosta veću tačnost od željene.

i	x[i]
0	**2**.50000000000
1	**2**.16417910448
2	**2.09**713535581
3	**2.09455**523239
4	**2.09455**148155
5	**2.0945514815**4

,

i	x[i]
0	**2**.50000000000
1	**2**.16417910448
2	**2**.11594357455
3	**2**.10151659905
4	**2.09**685714226
5	**2.09**531875986
6	**2.094**80725963
7	**2.094**63679621
8	**2.09457**994367
9	**2.09456**097750
10	**2.09455**464979
11	**2.09455**253861
12	**2.094551**83423
13	**2.094551**59921
14	**2.094551**52080

Uzgred, tačno realno rešenje jednačine $f(x) = 0$ je

$$x = \sqrt[3]{\frac{5}{2} + \frac{\sqrt{1929}}{18}} + \sqrt[3]{\frac{5}{2} - \frac{\sqrt{1929}}{18}} = 2.09455148... \ .$$

△

[10] Ovu jednačinu je Njutn upotrebio 1669. godine da bi ilustovao svoj metod.

Primer 4. ***(Izračunavanje n-tog korena broja)*** *Neka je $a > 0$. Konstruisati iterativni metod za izračunavanje $\sqrt[n]{a}$.*

Potom, izračunati $\sqrt[5]{13}$, sa tačnošću $\varepsilon = 10^{-8}$.

Rešenje. Definišimo funkciju $f_a : \mathbb{R} \to \mathbb{R}$ sa

$$f_a(x) = x^n - a, \quad n \in \mathbb{N},\ a > 0.$$

Rešenje jednačine (realno) $f(x) = 0$ je $\sqrt[n]{a}$. Lako nalazimo da je $f_a'(x) = nx^{n-1}$, $\forall x \in \mathbb{R}, \forall n \in \mathbb{N}$. Sada imamo da je $f_a'(x) \neq 0$ za svaku okolinu tačke $\sqrt[n]{a}$ koja ne sadrži nulu, pa možemo primeniti Njutnov metod za određivanje korena ove jednačine.

Iterativni proces se sada lako konstruiše.

$$x_{k+1} = x_k - \frac{f_a(x_k)}{f_a'(x_k)} = x_k - \frac{x_k^n - a}{nx_k^{n-1}} = \frac{1}{n}\left((n-1)x_k + \frac{a}{x_k^{n-1}}\right).$$

Primetimo da je konstruisani iterativni metod reda 2, zbog toga što je Njutnov metod reda 2.

Sada dajemo pseudokod ovog iterativnog algoritma.

Algoritam 6 Izračunavanje $n-$tog korena datog pozitivnog broja a

Input Pozitivan broj a, prirodan broj n, početna aproksimacija x_0 i tačnost ε.

1: $k := 1$
2: $x_1 := \frac{1}{n}\left((n-1)x_0 + \frac{a}{x_0^{n-1}}\right)$
3: **while** $|x_k - x_{k-1}| \geq \varepsilon$ **do**
3: $\quad$ pom $:= x_k$
4: $\quad x_k := \frac{1}{n}\left((n-1)\text{pom} + \frac{a}{\text{pom}^{n-1}}\right)$
5: $\quad x_{k-1} :=$ pom
6: $k := k + 1$
8: **end while**
7: **return** x_k

Izlazni kriterijum je $|x_n - x_{n-1}| < \varepsilon$. Sada lako možemo napraviti program u bilo kom jeziku za računanje n-tog korena sa željenom tačnošću.

```
NKoren[a_, n_, x0_, eps_] :=
 Module[{xnew, xold, pom, k, Tablica = {{"k", "x[k]"}}},
  k = 0;
  AppendTo[Tablica, {k, N[x0, 12]}];
  k++;
  xold = x0;
  k = 1;
  xnew = (1/n)*((n - 1) xold + a/(xold^(n - 1)));
  AppendTo[Tablica, {k, N[xnew, 12]}];
  k++;
  While[Abs[xnew - xold] >= eps && k <= 10,
   pom = xnew;
   xnew = (1/n)*((n - 1) pom + a/(pom^(n - 1)));
   AppendTo[Tablica, {k, N[xnew, 12]}];
```

```
  xold = pom;
  k++
  ];
 Return[TableForm[Tablica]]
 ]
```

Aproksimaciju vrednosti $\sqrt[5]{13}$ dobijamo pozivajući funkciju `NKoren` za početnu tačku $x_0 = 1$.

k	x[k]
0	**1**.00000000000
1	3.40000000000
2	2.73945618467
3	2.23773027445
4	**1**.89387553831
5	**1**.71720103369
6	**1.67**277294067
7	**1.6702**8508572
8	**1.6702776524**0
9	**1.67027765233**

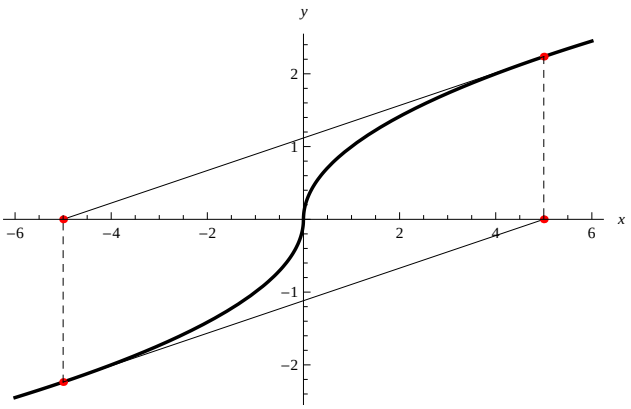
Primetimo da smo dobili približnu vrednost sa većom tačnošću nego što je bila željena tačnost. △

Sada ćemo dati jedan karakterističan primer gde Njutnov metod za neke startne vrednosti osciluje.

Primer 5. ***(Oscilovanje Njutnovog metoda)*** *Konstruisati funkciju, tako da za neku startnu vrednost iteracije Njutnovog metoda osciluju oko dve vrednosti.*

Rešenje. Definišimo funkciju $f : \mathbb{R} \to \mathbb{R}$ sa

$$f(x) = \begin{cases} \sqrt{x}, & x \geq 0 \\ -\sqrt{-x}, & x < 0 \end{cases}.$$



Za startnu vrednost $x_0 = 5$ dobijamo vrednosti vrednosti prikazane u narednoj tabeli a oscilovanje se jasno vidi na sledećem grafiku.

$$\left\{\begin{array}{cc} \text{i} & \text{x[i]} \\ 0 & 5. \\ 1 & -5. \\ 1 & 5. \\ 2 & -5. \\ 3 & 5. \\ 4 & -5. \\ 5 & 5. \end{array}\right\}$$

$\triangle$

2.6 Metod sečice

U svakoj iteraciji Njutnovog metoda potrebno je izračunati vrednost prvog izvoda $f'(x)$ u tački $x = x_n$. Ako je teško izračunati vrednost prvog izvoda, može se koristiti aproksimacija

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

Zamenom u formulu kojom je definisan Njutnov iterativni metod, dobijamo formulu kojom je definisan **metod sečice**:

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

Naziv metoda potiče iz činjenice da ako imamo dve startne vrednosti x_1 i x_2 narednu aproksimaciju dobijamo tako što konstruišemo pravu (sečicu) kroz tačke $(x_1, f(x_1))$ i $(x_2, f(x_2))$ i i odredimo je iz uslova preseka te prave i x-ose.

Sledeće teoreme daju uslove pod kojima je metod sečice konvergentan i red konvergencije metoda. Dokaz se može naći u [1].

Teorema 10. *Neka je $f \in \mathcal{C}^{(2)}(a, b)$, takva da je $f'(x) \geq m$ za neko $m > 0$ i za svako $x \in (a, b)$. Ako je $\overline{x}$ rešenje jednačine $f(x) = 0$, tada postoji $\varepsilon > 0$ tako da metod sečice konvergira ka $\overline{x}$ za svako $x_1, x_2 \in [\overline{x} - \varepsilon, \overline{x} + \varepsilon]$ i pritom važi*

$$|x_n - \overline{x}| \leq C q^{F_n},$$

gde su $C > 0$ i $q \in [0, 1)$ konstante a F_n je n-ti Fibonačijev[11] broj[12].

Metod sečice je najjednostavniji metod sa memorijom. Za izračunavanje svake naredne iteraciji je potrebno znati dve prethodne.

Teorema 11. *Uz pretpostavku da je $f''(\overline{x}) \neq 0$, metod sečice ima red konvergencije*

$$r = \frac{1 + \sqrt{5}}{2}$$

i asimptotsku konstantu greške

$$C = \left| \frac{f''(\overline{x})}{2f'(\overline{x})} \right|^{\frac{\sqrt{5}-1}{2}}.$$

[11] Leonardo Bonacci (1170-1250)-italijanski matematičar

[12] Fibonačijevi brojevi su elementi niza definisanog sa: $F_1 = 1$, $F_2 = 1$ i $F_n = F_{n-1} + F_{n-2}$, $n \geq 3$. Ovaj niz je jedan od najpoznatijih nizova u matematici i računarstvu a potiče iz mnogih pojava u prirodi.

2.6.1 Algoritam

Na osnovu izloženog lako konstruišemo algoritam koji je dat sledećim pseudokodom.

Algoritam 7 Metod sečica

Input Funkcija f, početne aproksimacije x_0 i x_1 i tačnost ε.

$k := 0$
xold1 $:= x_0$
$k := k + 1$
xold2 $:= x_1$
$k := k + 1$
$x_3 :=$ xold2 $- \frac{\text{xold2}-\text{xold1}}{(f(\text{xold2})-f(\text{xold1}))} f(\text{xold2})$
$k := k + 1$
while $|x_k -$ xold2$| \geq \varepsilon$ **do**
    xpom $:= x_k$
    $x_k := \frac{\text{xpom}-\text{xold1}}{(f(\text{xpom})-f(\text{xold1}))} f(\text{xpom})$
    xold1 := xold2
    xold2 := xpom
    $k := k + 1$
end while
return x_k

Primetimo da smo koristili pomoće promenljive xpom1, xpom2 i xpom.

2.6.2 Implementacija

Na osnovu izloženog pseudokoda lako implementiramo metod sečice.

```
MetodSecice[f_, x0_, x1_, eps_] :=
 Module[
  {xold1, xold2, xnew, xpom, i, Tabelica = {{"i", "x[i]"}}},
  i = 0;
  xold1 = x0;
  AppendTo[Tabelica, {i, N[xold1, 12]}];
  i++;
  xold2 = x1;
  AppendTo[Tabelica, {i, N[xold2, 12]}];
  i++;
  xnew = xold2 - ((xold2 - xold1)/(f[xold2] - f[xold1]))*f[xold2];
  AppendTo[Tabelica, {i, N[xnew, 12]}];
  i++;
  While[Abs[xnew - xold2] >= eps,
   xpom = xnew;
   xnew = xpom - ((xpom - xold1)/(f[xpom] - f[xold1]))*f[xpom];
   AppendTo[Tabelica, {i, N[xnew, 12]}];
   xold1 = xold2;
   xold2 = xpom;
   i++
```

```
 ];
Return[TableForm[Tabelica]]
]
```

Primetimo da implementacija nije baš najbolja jer u svakoj iteraciji izračunavamo dva puta jednu istu vrednost funkcije. Ovo se može ispraviti ako najpre zapamtimo izračunatu vrednost.

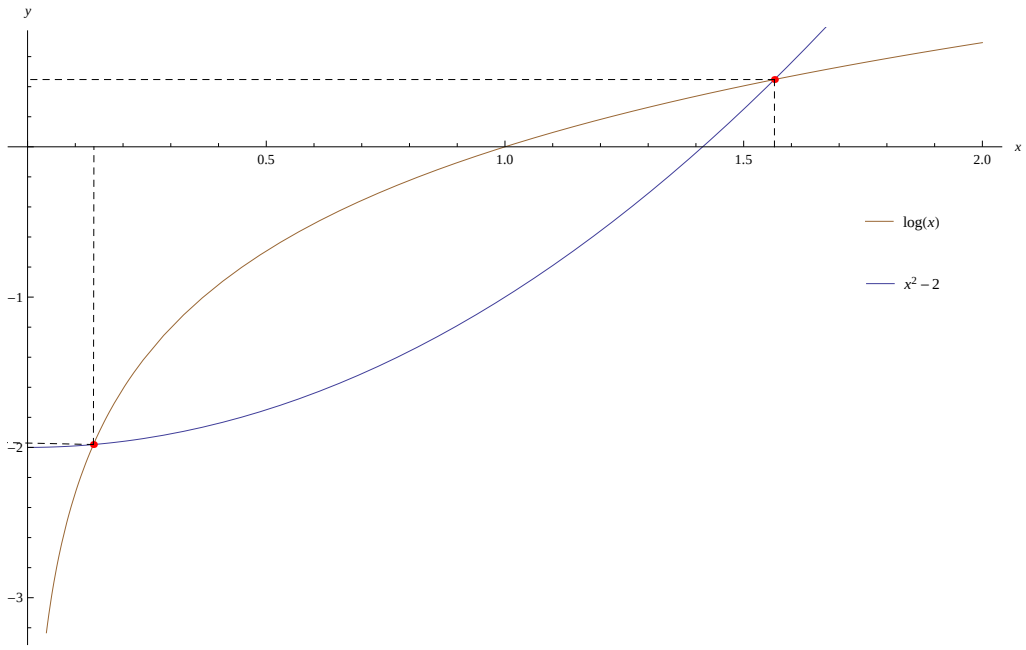
Ilustrujmo metod sečica sledećim primerom.

Primer 6. *Metodom sečica, sa tačnošću* $\varepsilon = 10^{-7}$*, naći sva rešenja jednačine* $x^2 - \ln x - 2 = 0$.

Rešenje. Zapišimo datu jednačinu u njenom ekvivalentnom obliku

$$x^2 - 2 = \ln x.$$

Sada skicirajući grafik funkcije na levoj strani jednakosti i skicirajući grafik funkcije na desnoj strani jednakosti možemo lokalizovati rešenja jednačine.



Sada lako zaključujemo da jednačina ima dva rešenja. Prvo rešenje se nalazi u segmenu $[0, 0.5]$ a drugo u segmentu $[1.5, 2]$. Za prvi koren startujemo sa $x_0 = 0.1$ i $x_1 = 0.3$, a za drugi koren sa $x_0 = 1.5$ i $x_1 = 2$. Dobijeni su sledeći rezulati:

i	x[i]
0	0.100000000000
1	0.300000000000
2	**0.1**61374695057
3	**0.1**41479157602
4	**0.13**5809265484
5	**0.13**8127024811
6	**0.13793**2168127
7	**0.137934**03435
8	**0.1379348255**82

,

i	x[i]
0	1.50000000000
1	2.00000000000
2	**1.5**5315708205
3	**1.564**83030991
4	**1.564**52490690
5	**1.56446191**452
6	**1.56446225920**
7	**1.56446225926**

U prvoj tabeli prikazane su aproksimacije manjeg a u drugoj većeg korena jednačine. Boldirane su tačne cifre. $\triangle$

2.7 Metod Regula-falsi

Kod metoda sečice može se desti da razlika $f(x_n) - f(x_{n-1})$ koja se nalazi u imeniocu može vrlo brzo postati bliska nuli. Zato se ovo opasno deljenje brojevima bliskim nuli može izbeći ako fiksiramo jednu početnu tačku kroz ceo iterativni proces.

2.7.1 Algoritam

Dakle, modifikacijom metoda sečice dobijamo iterativni proces

$$x_{n+1} = \frac{x_n - x_0}{f(x_n) - f(x_0)} f(x_n), \quad n \in \mathbb{N}.$$

Takođe, poželjno je da za sve naredne aproksimacije x_n gde je $n \in \mathbb{N}$ važi da su x_n i x_0 sa različitih strana korena jednačine $\overline{x}$. Ovako definisan metod je stabilniji od metoda sečice ali se odlikuje samo linearnom konvergencijom. Lako formiramo pseudokod ovog algoritma.

Algoritam 8 Metod Regula falsi

Input Funkcija f, početne aproksimacije $x_0, x_1 \in [a, b]$ i tačnost ε.
1: $n := 2$
2: $x_2 := x_0 - \frac{x_1 - x_0}{f(x_1) - f(x_0)}$;
3: **while** $|x_n - x_{n-1}| \geq \varepsilon$ **do**
3: $\quad$ pom $:= x_n$
4: $\quad x_n :=$ pom $- \frac{\text{pom} - x_0}{f(\text{pom}) - f(x_0)} f(\text{pom})$
5: $\quad x_{n-1} :=$ pom
6: $n := n + 1$
8: **end while**
7: **return** x_n

2.7.2 Implementacija

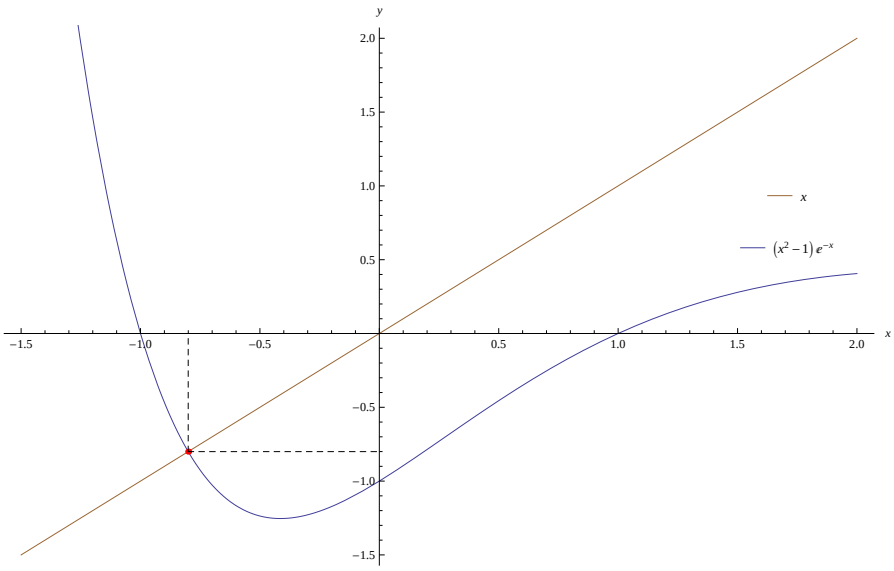
```
MetodRegulaFalsi[f_, x0_, x1_, eps_] :=
 Module[{Tablica = {{"i", "x[i]"}}, xold, xnew, xpom, i},
  xold = x0;
  i = 0;
  AppendTo[Tablica, {i, N[xold, 12]}];
  xold = x1;
  i = 1;
  AppendTo[Tablica, {i, N[xold, 12]}];
  i++;
  xnew = xold - ((xold - x0)/(f[xold] - f[x0]))*(f[xold]);
  AppendTo[Tablica, {i, N[xnew, 12]}];
  i++;
  While[Abs[xnew - xold] >= eps,
```

```
 xpom = xnew;
 xnew = xpom - ((xpom - x0)/(f[xpom] - f[x0]))*(f[xpom]);
 AppendTo[Tablica, {i, N[xnew, 12]}];
 xold = xpom;
 i++
 ];
Return[TableForm[Tablica]]
]
```

Primer 7. *Koristeći metod Regula-falsi, sa tačnošću* $\varepsilon = 10^{-4}$*, rešiti jednačinu*

$$x = (x^2 - 1)e^{-x}.$$

Rešenje. Skicirajmo najpre grafike funkcija na levoj i na desnoj strani jednakosti da bismo izvršili lokalizaciju nula jednačine.



Startujući sa $x_0 = -1$ i $x_1 = -0.5$ dobijamo sledeće aproksimacije:

i	x[i]
0	-1
1	-0.5
2	$-\mathbf{0}.712071$
3	$-\mathbf{0}.777261$
4	$-\mathbf{0}.794511$
5	$-\mathbf{0}.798869$
6	$-\mathbf{0}.799957$
7	$-\mathbf{0.800}228$
8	$-\mathbf{0.800}295$

Tačno rešenje jednačine je $x = -0.8003176317391357...$. $\triangle$

2.8 Konsrukcija metoda višeg reda ($r \geq 3$)

Sada ćemo pokušati da ubrazamo neke već izložene iterativne metode i da vidimo da li se to ubrzanje isplati, gledano u odnosu na to koliko će se povećati red konvergencije u odnosu na broj funkcijskih izračunavanja po iteraciji. Relevantni faktor će biti računska efikasnost iterativnog procesa koju smo već uveli.

2.8.1 Algoritam

Najpre dajemo sledeću teoremu.

Teorema 12. *Neka je*

$$x_{k+1} = \Phi(x_k), \quad k \in \mathbb{N} \cup \{0\},$$

iterativni proces sa konvergencijom reda r, gde je Φ funkcija koja je $(r+1)$-puta neprekidno diferencijabilna u nekoj okolini granične tačke $\overline{x}$ i neka je $\Phi'(\overline{x}) \neq r$. Tada je sa

$$x_{k+1} = x_k - \frac{x_k - \Phi(x_k)}{1 - \frac{1}{r}\Phi'(x_k)}$$

definisan iterativni proces reda najmanje $r+1$.

Dokaz ove teoreme se može naći u [2].

Primenimo sada prethodnu teoremu na ubrzavanje Njutnovog metoda.

Iterativna funkcija Njutnovog metoda je

$$\Phi(x) = x - \frac{f(x)}{f'(x)}.$$

Koristeći navedenu teoremu lako dobijamo metod

$$x_{k+1} = x_k - \frac{2f(x_k)f'(x_k)}{2((f'(x_k))^2 - f(x_k)f''(x_k)}.$$

Kako imamo ukupno $\theta = 3$ funkcijskih izračunavanja po svakoj iteraciji a metod je reda $r = 3$ zaključujemo da je računska efiksanost za ovaj iterativni metod $E_c = 3^{\frac{1}{3}} \approx 1.4422 > \sqrt{2}$. Dakle, zaključujemo da je ovako konstruisan iterativni metod blago efikasniji od Njutnovog ali nije primer optimalnog metoda, jer ne zadovoljava Kung-Traubovu hipotezu.

2.8.2 Implementacija

Kod implementacije, ako bi direktno sračunavli po datoj formuli, imali bi jedni izračunavanje više, jer treba 2 puta da se sračuna vrednost prvog izvoda u istoj tački. Zato, ovaj problem rešavamo tako što prvo zapamtimo vrednost prvog izvoda u nekoj pomoćnoj promenljivoj, a potom koristimo vrednost te pomoćne promenljive.

```
UbrzanNjutnovMetod[f_, x0_, eps_] :=
 Module[{Tablica = {{"i", "x[i]"}}, xold, prv, xnew, izv, xpom, i},
  xold = x0;
  i = 0;
```

Algoritam 9 Ubrzan Njutnov metod

Input Funkcija f, segment $[a, b]$ gde je izolovan koren jednačine, početna aproksimacija $x_0 \in [a, b]$ i tačnost ε.

1: $n := 1$
2: $x_1 := x_0 - 2\frac{f(x_0)f'(x_0)}{2((f'(x_0))^2 - f(x_0)f''(x_0)}$;
3: **while** $|x_n - x_{n-1}| \geq \varepsilon$ **do**
3: $\quad$ pom $:= x_n$
4: $\quad x_n :=$ pom $- 2\frac{f(\text{pom})f'(\text{pom})}{2((f'(\text{pom}))^2 - f(\text{pom})f''(\text{pom})}$
5: $\quad x_{n-1} :=$ pom
6: $n := n + 1$
8: **end while**
7: **return** x_n

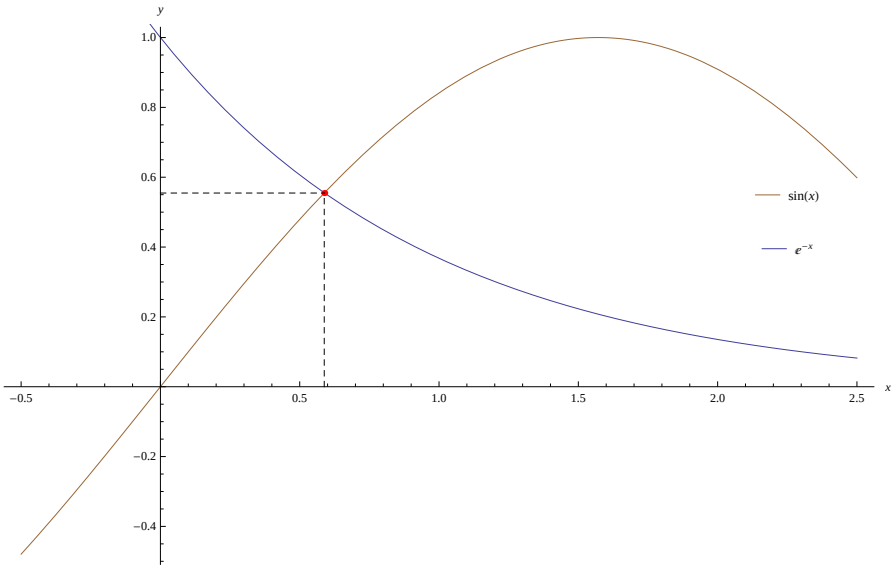
```
AppendTo[Tablica, {i, N[xold, 12]}];
i++;
prv = f'[x0];
xnew = xold - 2 (f[x0] prv)/(2 (prv)^2 - f[x0] f''[x0]);
AppendTo[Tablica, {i, N[xnew, 12]}];
i++;
While[Abs[xnew - xold] >= eps,
 xpom = xnew;
 izv = f'[xpom];
 xnew = xpom - 2 (f[xpom] izv)/(2 (izv)^2 - f[xpom] f''[xpom]);
 AppendTo[Tablica, {i, N[xnew, 12]}];
 xold = xpom;
 i++
 ];
Return[TableForm[Tablica]]
]
```

Primer 8. *Naći najmanji koren jednačine Njutnovim metodom i Ubrzanim Njutnovim metodom, sa tačnošću $\varepsilon = 10^{-6}$. Uporediti dobijene rezultate.*

Rešenje. Jednačinu možemo transformisati u njen ekvivalentan oblik

$$\sin x = e^{-x}.$$

Skiciranjem grafika funkcija $y = \sin x$ i $y = e^{-x}$ možemo lokalizovati rešenje.



Startujući sa $x_0 = 0.5$ dobijeni su sledeći rezultati.

$$\left\{ \begin{pmatrix} \text{i} & \text{x[i]} \\ 0 & \mathbf{0.5}000000000 \\ 1 & \mathbf{0.58}56438170 \\ 2 & \mathbf{0.5885}294126 \\ 3 & \mathbf{0.5885327}440 \\ 4 & \mathbf{0.5885327}440 \end{pmatrix}, \begin{pmatrix} \text{i} & \text{x[i]} \\ 0 & \mathbf{0.5}000000000 \\ 1 & \mathbf{0.588}4141572 \\ 2 & \mathbf{0.5885327}440 \\ 3 & \mathbf{0.5885327}440 \end{pmatrix} \right\}$$

Uprvoj matrici u listu su prikazani rezultati za Njutnov metod a u drugoj za Ubrzani Njutnov metod. Primetimo da je Ubrzani Njutnov metod za jednu iteraciju brži.

Tačno rešenje jednačine je $x = 0.5885327439818611...$. $\triangle$

2.9 Metod Ostrovskog

Sada dajemo primer optimalnog metoda četvrtog reda. Metod je definisan sa

$$y_n = x_n - \frac{f(x_n)}{f'(x_n)}, \quad x_{n+1} = y_n - (x_n - y_n) \cdot \frac{f(y_n)}{f(x_n) - 2f(y_n)}.$$

Metodi definisani na ovaj način nazivaju se **višetačkasti** metodi. Zapravo, metod $y_n = g_1(x_n)$, $x_{n+1} = g_2(x_n, y_n)$ ekvivalentan je metodu $x_{n+1} = g(x_n)$, gde je $g(x) = g_2(x, g_1(x))$.

Ukupno imamo 3 funkcijskih izračunavanja po svakoj iteraciji pa imamo da metod zadovoljava Kung-Traubovu hipotezu. Računska efiksandost metoda je $E_c = 4^{\frac{1}{3}} \approx 1.5874$. Dakle računska efikasnost metoda je veća i od efikasnosti Njutnovog i Ubrzanog Njutnovog metoda.

2.9.1 Implementacija

```
Ostrovski[f_, x0_, eps_] :=
 Module[{Tablica = {{"i", "x[i]"}}, i, xold, xnew, xpom, yold, ypom,
```

```
 ynew, pom1, pom2},
i = 0;
AppendTo[Tablica, {i, x0}];
xold = x0;
yold = xold - f[xold]/f'[xold];
pom1 = f[yold];
xnew = yold - (xold - yold)*(pom1/(f[xold] - 2 pom1));
i = 1;
AppendTo[Tablica, {i, xnew}];
While[Abs[xnew - xold] >= eps,
 xpom = xnew;
 ypom = ynew;
 ynew = xpom - f[xpom]/f'[xpom];
 pom2 = f[ynew];
 xnew = ynew - (xpom - ynew)*(pom2/(f[xpom] - 2 pom2));
 i++;
 AppendTo[Tablica, {i, xnew}];
 xold = xpom;
 yold = ypom;
 ];
Return[TableForm[Tablica]]
]
```

Primer 9. *Rešiti, sa tačnošću* $\varepsilon = 10^{-10}$ *jednačinu*

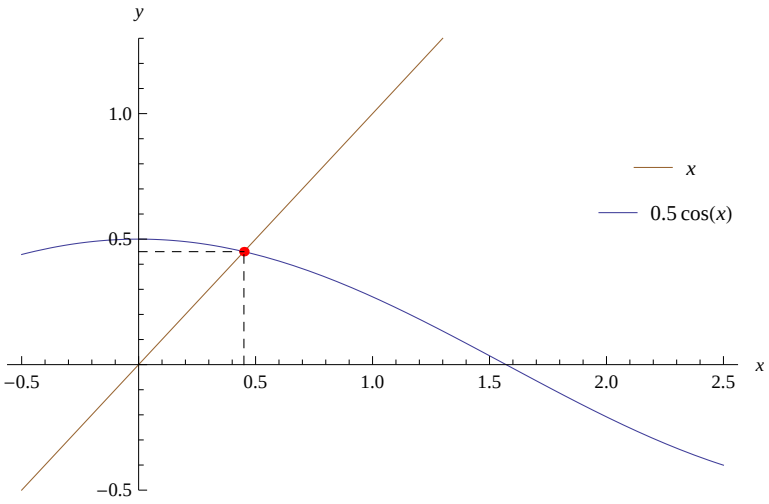
$$x = \frac{1}{2}\cos x,$$

Metodom Ostrovskog, Njutna, Ubrzanim Njutnovim metodom i metodom sečica i uporediti dobijene rezultate.

Rešenje. Skicrajmo najpre grafike funkcija

$$y = x, \quad y = \frac{1}{2}\cos x,$$

da bismo lokalizovali nulu funkcije. Početne aproksimacije su $x_0 = 0.2$ a za metod sečica $x_0 = 0.2$ i $x_1 = 0.5$. Dobijeni su sledeći rezultati. U prvoj tabeli su izračunavanja vezana za Njutnov metod, u drugoj za Ubrzan Njutnov metod, u trećoj za metod sečica a u četvrtoj za metod Ostrovskog.



i	x[i]
0	0.200000000000
1	**0.4**63826201474
2	**0.450**217695230
3	**0.450183611**510
4	**0.450183611295**
5	**0.450183611295**

,

i	x[i]
0	0.200000000000
1	**0**.200000000000
2	**0.450183611**229
3	**0.450183611295**

,

i	x[i]
0	0.200000000000
1	0.500000000000
2	**0.4**47720900976
3	**0.450**307528253
4	**0.450**184732831
5	**0.450183610**784
6	**0.450183611295**
7	**0.450183611295**

,

i	x[i]
0	0.200000000000
1	**0.450**238645547
2	**0.450183611295**

Tačno rešenje jednačine je $x = 0.45018361129487383...$. △

2.10 Algebarske jednačine

Kako smo ranije napomenuli, algebarske (polinomske) jednačine izučavamo odvojeno zbog njihovog značaja.

Definicija 5. *Jednačina oblika*

$$p(z) = a_0 z^n + a_1 z^{n-1} + ... + a_{n-1} z + a_n, \quad a_0, a_1, ..., a_n \in \mathbb{C}, \ a_0 \neq 0,$$

naziva se ***algebraska*** *ili* ***polinomska*** *jednačina stepena* $n \in \mathbb{N}$.

Navodimo, bez dokaza fundamentalnu teoremu matematike.

Teorema 13. ***(Osnovna teorema Algebre)*** *Svaki polinom* $p \in \mathbb{C}[x]$ *ima barem jednu nulu* $z \in \mathbb{C}$.

Direktna posledica osnovne teoreme Algebre je sledeće značajno tvrđenje.

Posledica 2. *Svaki polinom* $p \in \mathbb{C}[x]$ *stepena* n *ima* n *kompleksnih nula, računajući svaku onoliko puta kolika je njena višestrukost.*

Nule polinoma zavise direktno od koeficijenata. Iz Algebre je poznato sledeće takođe fundamentalno tvrđenje.

Teorema 14. *Algebarsku jednačinu stepena većeg od 4, u opštem slučaju, nije moguće rešiti tako da rešenja budu izražena preko njenih koeficijenata i matematičkih operacija sabiranja, množenja i korenovanja u zatvorenom obliku.*

Moguće je na izvestan način povezati nule polinoma i njegove koeficijente. O tome govori naredna teorema.

Teorema 15. ***(Vietove[13] formule)*** *Neka su $x_1, ..., x_n$ nule polinoma $p \in \mathbb{C}[x]$. Tada važe sledeće formule*

$$x_1 + ... + x_n = -\frac{a_1}{a_0},$$

$$x_1x_2 + x_1x_3 + ... + x_{n-1}x_n = \frac{a_2}{a_0},$$

$$...$$

$$\sum_{1 \le i_1 < i_2 < ... < i_k \le n} x_{i_1}x_{i_2} \cdots x_{i_k} = (-1)^k \frac{a_k}{a_0},$$

$$x_1x_2 \cdots x_n = (-1)^n \frac{a_n}{a_0}.$$

Opširnije o ovim teoremama može se naći u svim standardnim udžbenicima iz Algebre, videti na primer [16].

Mi smo do sada razmatrali iterativne metode za određivanje jednog korena jednačine. Sada lako možemo konstruisati algoritam za određivanje svih korena algebarskih jednačina, koristeći se izloženim teoremama.

Naime, nekom metodom odredimo jednan koren jednačine. Neka je to $z = z_0$. Podelimo našu jednačinnu sa $z - z_0$ i ponovimo prethodni korak nad dobijenim polinomom. Radi preglednosti dajemo pseudokod ovog algoritma.

Algoritam 10 Izračunavanje svih nula polinoma

Input Polinom $p(z)$ stepena $n \in \mathbb{N}$.
1: $p_1(z) := p(z)$
2: $i := 1$
3: **while** $\deg p_i(z) > 0$ **do**
4: Izračunati nulu x_i polinoma $p_i(z)$ primenom nekog metoda za računanje jedne nule.
5: **if** $z_i \in \mathbb{R}$ **then**
6: $p_{i+1}(z) := p_i(z)\text{div}(z - z_i)$
7: $i := i + 1$
8: **else**
9: $r_i := 2\text{Re}(z_i)$
10: $s_i := -|z_i|^2$
11: $p_{i+2}(z) := p_i(z)\text{div}(z^2 - r_iz - s_i)$
12: $z_{i+1} := \overline{z_i}$
13: $i := i + 2$
14: **end if**
15: **end while**
16: **return** $\{x_1, ..., x_n\}$

Međutim, dati algoritam se ne primenjuje u praksi zato što se vrlo brzo ispolji **nagomilavanje računskih grešaka**. Zbog toga, razvijaju se metodi za **simultano** (istovremeno) određivanje svih nula polinoma.

[13] Franciscus Vieta (1540-1603)-francuski matematičar

2.10.1 Lokalizacija nula polinoma

Da bismo razvili uopšte neki simultani metod, moramo znati u kojoj oblasti se nalaze nule funkcije, jer je dobar odabir početnih aproksimacija vrlo bitna stvar koja će se odraziti kroz ceo iterativni proces. Kako je u ovom slučaju pomenuta funkcija polinomska, možemo vrlo lako odrediti samo na osnovu njegovih koeficijenata oblast u $\mathbb{C}$ u kojoj su smeštene sve njegove nule.

Mi ćemo dokazati samo najosnovniju teoremu ovog tipa, jer je koristeći ideju njenog dokaza moguće dokazati čitav niz ovakvih teorema koje daju bolja ograničenja pomenute oblasti. Za više informacija pogledati izvanrednu zbirku zadataka [10].

Teorema 16. *Neka je data algebarska jednačina*

$$p(z) = a_0 z^n + a_1 z^{n-1} + ... + a_{n-1} z + a_n, \quad a_0, a_1, ..., a_n \in \mathbb{C},\ a_0 \neq 0,$$

i $a = \max\{|a_0|, |a_1|, ..., |a_{n-1}|\}$ *i neka je* $A = \max\{|a_1|, |a_1|, ..., |a_n|\}$. *Tada sve nule* $z_i,\ i \in \{1, ..., n\}$ *polinoma* p *se nalaze u prstenu*

$$\frac{|a_n|}{a + |a_n|} \leq |z_i| < \frac{|a_0| + A}{|a_0|}.$$

Dokaz. Kako je $\frac{|a_0|+A}{|a_0|} > 1$, dovoljno je posmatrati one korene polinoma koji su po modulu veći od 1. Tada je

$$\begin{aligned} p(z) &\geq |a_0 z^n| - \left(|a_1 z^{n-1}| + |a_2 z^{n-2}| + ... + |a_n|\right) \\ &\geq |a_0 z^n| - A\left(|z^{n-1}| + |z^{n-2}| + ... + |z| + 1\right) \\ &= |a_0||z|^n - A\frac{|z|^n - 1}{|z| - 1} > \left(|a_0| - \frac{A}{|z| - 1}\right)|z|^n. \end{aligned}$$

Prethodna nejednakost važi za svako $z \in \mathbb{C}$. Neka je $z = z_i,\ i \in \{1, ..., n\}$ proizvoljna nula polinoma p. Kada bi

$$\left(|a_0| - \frac{A}{|z_i| - 1}\right)|z_i|^n \geq 0$$

tada bi $|p(z_i)| > 0$ pa z_i ne bi bila nula polinoma što je kontradikcija. Dakle, ako je $z = z_i$ nula polinoma, tada mora da važi

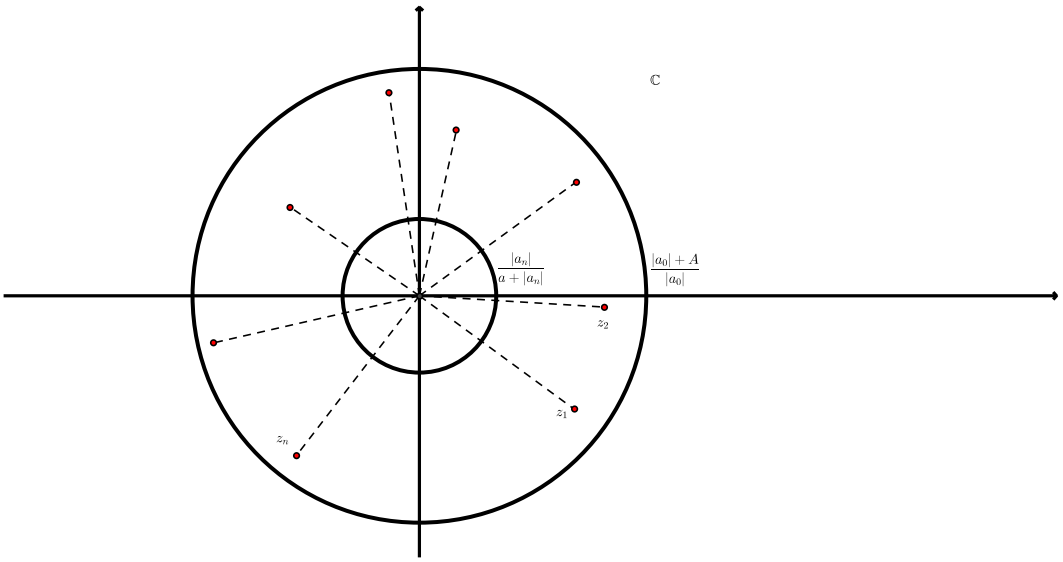
$$\left(|a_0| - \frac{A}{|z_i| - 1}\right)|z_i|^n < 0.$$

Poslednja nejednakost je ekvivalentna sa

$$|z_i| < \frac{|a_0| + A}{|a_0|}.$$

Levu stranu nejednakosti dobijamo kada upravo dokazanu desnu stranu nejednakosti primenimo na polinom $q(z) = z^n p\left(\frac{1}{z}\right)$. □

Dakle, oblast je kružni prsten u $\mathbb{C}$ sa centrom u koordinatnom početku.



2.10.2 Vajerštrasov metod

Osnovni simultani metod za izračunavanje svih nula polinoma je Vajerštrasov metod. Ovaj metod je u literaturi poznat kao i Durand-Kernerov metod. Više autora je do ovog metoda dolazilo nezavisno jedan od drugog, što potvrđuje važnost dobijanja ovakvih iteratvnih metoda.

Ovaj metod je samo primenljiv na polinome koji nemaju višestrukih nula. Pođimo od jednakosti

$$p(z) = a_0(z - z_1)(z - z_2) \cdots (z - z_n).$$

Ako izrazimo nulu $z_i,\ i \in \{1, ..., n\}$ dobijamo jednakost

$$z_i = z - -\frac{p(z)}{a_0 \prod_{\substack{j=1 \\ j \neq i}}^{n} (z - z_j)}.$$

Neka su $z_1^{(k)}, z_2^{(k)}, ..., z_n^{(k)}$ aproksimacije korena polinoma u k-toj iteraciji. Prethodni izraz nam daje motivaciju za definisanje iterativnog metoda

$$z_i^{(k+1)} = z_i^{(k)} - \frac{p\left(z_i^{(k)}\right)}{a_0 \prod_{\substack{j=1 \\ j \neq i}}^{n} \left(z_i^{(k)} - z_j^{(k)}\right)}.$$

Ukoliko su startne vrednosti $z_1^{(k)}, z_2^{(k)}, ..., z_n^{(k)}$ blizu tačnih vrednosti $z_1, z_2, ..., z_n$ Vajerštrasov metod ima kvadratnu konvergenciju.

Zanimljiva je sledeća hipoteza.

Hipoteza 2. *Vajreštrasov metod je globalno konvergentan.*

U praksi, na računarima, hipoteza još nije opovrgnuta nekim kontra primerom. Zanimljivo je i sledeće tvrđenje.

Teorema 17. *Za svako $k \in \mathbb{N}$ važi*

$$\sum_{i=1}^{n} z_i^{(k)} = \sum_{i=1}^{n} z_i = -a_1.$$

Dokaz ove teoreme koristi metode Kompleksne analize pa ćemo ga izostaviti. Dokaz se može naći u [2].

Sledi implementacija Vajerštrasovog metoda izvedena u programskom paketu `Mathematica`.

Korišćena je teorema o lokalizaciji nula polinoma, za odabir početne aproksimacije.

Izlazni kriterijum je dat funkcijom `MxGreska`. Naime, u svakoj iteraciji se sračunava matrica dimenzija $n \times 1$ u kojoj su smeštene aproslimacije nula polinoma. Za svaku sračunatu matricu se izračunava vrednost polinoma u svakoj tački iz matrice. Potom se uzima najveća vrednost po modulu i funkcija `MxGreska` vraća tu vrednost. Izračunavanja prekidamo kada je vrednost funkcije `MxGreska` u nekoj iteraciji manja od unete tačnosti $\varepsilon > 0$.

Kako početnu vrednost biramo koristeći ugrađenu funkciju `RandomComplex` ograničili smo broj iteracija sa veličinom `maxIter` = 10000 jer nismo sigurni da li iterativni metod konvergira.

Funkcija `VajerstrasovMetod` vraća listu u kojoj su smeštene aproksimacije nula dobijenih pomoću ovog algoritma na prvoj poziciji a na drugoj broj iteracija koje je algoritam izvršio.

```
MxGreska[pol_, mat_] := Module[{l = {}, i, mx, br},
  For[i = 1, i <= Dimensions[mat][[1]], i++,
   AppendTo[l, pol /. Variables[pol][[1]] -> mat[[i, 1]]]];
  mx = Max[Abs[l]];
  Return[mx]
  ]

Vajerstras[pol_, eps_] :=
 Module[
  {x, pol1, an, w, n, i, j, k, l, p, s, a0, Xnew, Xold, mat, Xpom, A,
   a, grD, grL, mid, maxIter = 10000},
  x = Variables[pol][[1]];
  l = CoefficientList[pol, x];
  n = Length[l] - 1;
  a0 = l[[-1]];
  an = l[[1]];
  pol1 = pol/a0;
  a = Max[Take[Abs[l], -Length[l] + 1]];
  A = Max[Take[Abs[l], Length[l] - 1]];
  grD = (Abs[a0] + A)/(Abs[a0]);
  grL = (Abs[an])/(a + Abs[an]);
  mid = 0.5*(grL + grD);
  mat = RandomReal[{-1, 1}, {n, 1}];
  Xold = RandomReal[{0, 1}, {n, 1}];
  For[w = 1, w <= n, w++,
   Xold[[w, 1]] = RandomComplex[{-(1 + I)*mid, (1 + I)*mid}]];
```

```
For[i = 1, i <= n, i++,
 mat[[i, 1]] = (pol1 /. x -> Xold[[i, 1]])/(Product[
     If[i != j, (Xold[[i, 1]] - Xold[[j, 1]]), 1], {j, 1, n}])];
Xnew = Xold - mat;
k = 1;
While[Abs[MxGreska[pol1, Xnew]] >= eps && k <= maxIter,
 Xpom = Xnew;
 For[p = 1, p <= n, p++,
  mat[[p, 1]] = (pol1 /. x -> Xpom[[p, 1]])/(Product[
      If[p != s, (Xpom[[p, 1]] - Xpom[[s, 1]]), 1], {s, 1, n}])];
 Xnew = Xpom - mat;
 Xold = Xpom;
 k++];
Return[{Xnew, k}]
]
```

Primer 10. *Odrediti Vajerštrasovim algoritmom, sa tačnošću* $\varepsilon = 10^{-15}$ *sve nule polinoma*

$$p(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1.$$

Rešenje. Pokažimo najpre da dati polinom nema višestrukih korena, da bismo uopšte mogli da primenimo Vajerštrasov metod. Kako za svako $x \in \mathbb{C} \setminus \{1\}$ važi

$$p(x) = \frac{x^7 - 1}{x - 1},$$

a rešenja racionalne funkcije sa desne strane jednakosti su **koreni iz jedinice**[14], imamo da polinom p nema višestrukih korena.

Pozivom `VajerstrasovMetod`[1+x+x^2+x^3+x^4+x^5+x^6, 10^(-15)] dobijamo rezultat

$$\left\{ \begin{pmatrix} -0.2225209339563144 + 0.9749279121818236i \\ 0.6234898018587336 + 0.7818314824680298i \\ -0.9009688679024191 + 0.4338837391175581i \\ 0.6234898018587335 - 0.7818314824680298i \\ -0.22252093395631442 - 0.9749279121818236i \\ -0.9009688679024191 - 0.4338837391175581i \end{pmatrix}, 11 \right\}.$$

Broj izvrešenih iteracija je 11 što je izuzetno dobro u odnosu na visoku zahtevanu tačnost. △

Pokušajmo sada da iskoristimo i izvod polinoma, da bi ubrzali Vajerštrasov metod. Neka je dakle polinom

$$p(z) = (z - z_1)(z - z_2) \cdots (z - z_n),$$

polinom koji ima sve proste nule. Sada imamo da je

$$\frac{p'(z)}{p(z)} = \sum_{i=1}^{n} \frac{1}{z - z_i} = \frac{1}{z - z_i} + \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{1}{z - z_j}.$$

[14]Koreni iz jedinice su kompleksni brojevi koji su rešenja jednačine $x^n = 1$. Oni čine temena pravilnog n-tougla koji je upisan u jediničnu kružnicu, čije je jedno teme kompleksan broj $\cos \frac{2\pi}{n} + i \sin \frac{2\pi}{n}$.

Ovaj izraz nam daje motivaciju za uvođenje iterativnog metoda:

$$z_i^{(k+1)} = z_i^{(k)} - \frac{1}{\frac{p'\left(z_i^{(k)}\right)}{p\left(z_i^{(k)}\right)} - \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{1}{z_i^{(k)} - z_j^{(k)}}}.$$

Implementacija ovog algoritma je slična implementaciji Vajerštrasovog algoritma.

```
MxGreska[pol_, mat_] := Module[{l = {}, i, mx, br},
  For[i = 1, i <= Dimensions[mat][[1]], i++,
   AppendTo[l, pol /. Variables[pol][[1]] -> mat[[i, 1]]]];
  mx = Max[Abs[l]];
  Return[mx]
  ]

Mealy[pol_, eps_] :=
 Module[
  {x, an, w, n, i, j, k, l, p, s, a0, Xnew, Xold, mat, Xpom, A, a,
   grD, grL, mid, izvodni, pol1, maxIter = 1000},
  x = Variables[pol][[1]];
  l = CoefficientList[pol, Variables[pol][[1]]];
  n = Length[l] - 1;
  a0 = l[[-1]];
  pol1 = pol/a0;
  an = l[[1]];
  a = Max[Take[Abs[l], -Length[l] + 1]];
  A = Max[Take[Abs[l], Length[l] - 1]];
  grD = (Abs[a0] + A)/(Abs[a0]);
  grL = (Abs[an])/(a + Abs[an]);
  mid = 0.5*(grL + grD);
  mat = RandomReal[{-1, 1}, {n, 1}];
  Xold = RandomReal[{0, 1}, {n, 1}];
  izvodni = D[pol1, Variables[pol][[1]]];
  For[w = 1, w <= n, w++,
   Xold[[w, 1]] = RandomComplex[{-(1 + I)*mid, (1 + I)*mid}]];
  For[i = 1, i <= n, i++,
   mat[[i, 1]] =
    1/((izvodni /. x -> Xold[[i, 1]])/(pol1 /. x -> Xold[[i, 1]]) -
       Sum[If[i != j, 1/(Xold[[i, 1]] - Xold[[j, 1]]), 0], {j, 1,
         n}])];
  Xnew = Xold - mat;
  k = 1;
  While[Abs[MxGreska[pol1, Xnew]] >= eps && k <= maxIter,
   Xpom = Xnew;
   For[p = 1, p <= n, p++,
    mat[[p, 1]] =
     1/((izvodni /. x -> Xpom[[p, 1]])/(pol1 /. x -> Xpom[[p, 1]]) -
        Sum[If[p != s, 1/(Xpom[[p, 1]] - Xpom[[s, 1]]), 0], {s, 1,
          n}])];
   Xnew = Xpom - mat;
   Xold = Xpom;
   k++];
```

```
Return[{Xnew, k}]
]
```

Primer 11. *Rešiti prethodni primer novodobijenim metodom i uporediti rezultate.*

Rešenje. Pozivom `Mealy`[1+x+x^2+x^3+x^4+x^5+x^6, 10^(-15)] dobijamo rezultat

$$\left\{\begin{pmatrix} -0.9009688679024191 + 0.4338837391175582i \\ -0.22252093395631442 - 0.9749279121818236i \\ 0.6234898018587336 + 0.7818314824680298i \\ 0.6234898018587335 - 0.7818314824680298i \\ -0.9009688679024191 - 0.4338837391175582i \\ -0.22252093395631442 + 0.9749279121818236i \end{pmatrix}, 7\right\}.$$

Vidimo da smo i u ovom slučaju postigli željenu tačnost, ali sa 7 iteracija, te možemo zaključiti da smo izvršili ubrzanje iterativnog Vajerštrasovog metoda. △

U literaturi je ovakav metod specijalan slučaj Melijevog metoda. Videti za više informacija [14] i [15].

3 Sistemi nelinearnih jednačina

U prethodnom poglavlju smo razmatrali numeričko rešavanje nelinarnih jednačina. Dakle, imali smo jednodimenzionalni problem. Problem se znato komplikuje ako ga proširimo na više dimenzija.

Objasnimo o čemu se radi, radi jednostavnosti, u dvodimenzionalnom slučaju. Neka su $f, g : D \to \mathbb{R}$ gde je $D \subset \mathbb{R}^2$. Sistem od dve linearne jednačine sa dve nepoznate je sistem

$$(\mathcal{S}_2) : \begin{cases} f(x, y) = 0 \\ g(x, y) = 0. \end{cases}$$

Dakle, u $\mathbb{R}^2$ ravni imamo dve krive i tražimo sve njihove preseke u oblasti $D \subset \mathbb{R}^2$.

3.1 Metod Njutna-Kantoroviča

Najveći problem je to što su jednačine **nelinearne**. Dakle, moramo nekako izvršiti njihovu **linearizaciju**. Posmatrajmo sistem od $n \in \mathbb{N}$ nelinearnih jednačina efinisan sa

$$(\mathcal{S}_n) : \begin{cases} f_1(x_1, x_2, ..., x_n) = 0 \\ f_2(x_1, x_2, ..., x_n) = 0 \\ \qquad\qquad ... \\ f_n(x_1, x_2, ..., x_n) = 0. \end{cases}$$

Neka je $\overrightarrow{x} = [x_1\ x_2 \cdots x_n]^T$ vektor promenljivih a $\overrightarrow{f} = [f_1\ f_2 \cdots f_n]^T$ vektor funkcija. Sada sistem (S_n) možemo zapisati u vektorskom obliku

$$\overrightarrow{f}(\overrightarrow{x}) = \overrightarrow{0},$$

gde je $\overrightarrow{0} = [\underbrace{0\ 0 \cdots 0}_{n}]^T$.

Neka je

$$\overrightarrow{x}^{(k)} = \left[x_1^{(k)}\ x_2^{(k)} \cdots x_n^{(k)}\right]^T$$

vektor promenljivih u k-toj iteraciji.

Na osnovu pretpostavke o diferencijabilnosti funkcija f_i možemo izvršiti linearnu aproksimaciju u okolini tačke $\overrightarrow{x}^{(k)}$ dobijamo

$$f_i\left(\overrightarrow{x}\right) = f\left(\overrightarrow{x}^{(k)}\right) + \frac{\partial f_i}{\partial x_1}\left(x_1 - x_1^{(k)}\right) + \frac{\partial f_i}{\partial x_2}\left(x_2 - x_2^{(k)}\right) + ... + \frac{\partial f_i}{\partial x_n}\left(x_n - x_n^{(k)}\right) + r_i^{(k)}.$$

Parcijalni izvodi su računati u tački $x = \overrightarrow{x}^{(k)}$. Neka je $\overrightarrow{\overline{x}} = [\overline{x_1}\ \overline{x_2} \cdots \overline{x_n}]^T$ tačno rešenje sistema (S_n). Zapisano u vektorskom obliku dobijamo jednačinu

$$0 = \overrightarrow{f}\left(\overrightarrow{x}^{(k)}\right) + J\left(\overrightarrow{x}^{(k)}\right)\left(\overrightarrow{\overline{x}} - \overrightarrow{x}^{(k)}\right) + r^{(k)},$$

gde je J**Jakobijeva** [15] **matrica** definisana sa

$$J(\overrightarrow{x}) = \left[\frac{\partial f_i}{\partial x_j}(\overrightarrow{x})\right]_{1\le i,j\le n},$$

i

$$r^{(k)} = \left[r_1^{(k)}\ r_2^{(k)} \cdots r_n^{(k)}\right].$$

Pod pretpostavkom da je matrica $J\left(\overrightarrow{x}^{(k)}\right)$ invertibilna dobijamo

$$\overrightarrow{\overline{x}} = \overrightarrow{x}^{(k)} - J^{-1}(\overrightarrow{x}^{(k)})\overrightarrow{f}(x^{(k)}) - J^{-1}(\overrightarrow{x})r^{(k)}.$$

Zanemarivanjem poslednjeg člana dobijamo iterativni metod

$$\overrightarrow{x}^{(k+1)} = \overrightarrow{x}^{(k)} - J^{-1}(\overrightarrow{x}^{(k)})\overrightarrow{f}(x^{(k)}), \quad k \in \mathbb{N} \cup \{0\}.$$

Zaustavni kriterijum će biti norma razlike uzastopnih iteracija. Kako su iteracije vektori-kolone koristićemo **Frobeniusovu**[16] normu. Frobeniusova norma za matricu $A \in \mathbb{C}^{n\times m}$ definisana je sa

$$||A||_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}|a_{ij}|^2}.$$

U programskom paketu `Mathematica` postoji ugrađena funkcija za normu vektora. Za matricu $A \in \mathbb{C}^{n\times m}$, Frobeniusova norma poziva se sa `Norm`$[A,$"`Frobenius`"$]$. Više o normama vektora i matrica videti u [17].

Konstruisani metod se naziva **metod Njutna-Kantoroviča**[17]. Uslove za konvergenciju je dao Kantorovič.

[15] Carl Gustav Jacob Jacobi (1804-1851)-nemački matematičar
[16] Ferdinand Georg Frobenius (1849-1917)-nemački matematičar
[17] Leonid Vitaliyevich Kantorovich (1912-1986)-ruski matematičar

Algoritam 11 Metod Njutna-Kantoroviča

Input Sistem nelinarnih jednačina, početni vektor $\overrightarrow{x}^{(0)}$ i tačnost $\varepsilon > 0$.

1: $i := 1$
2: $\overrightarrow{x}^{(1)} = \overrightarrow{x}^{(0)} - J^{-1}(\overrightarrow{x}^{(0)})\overrightarrow{f}\left(\overrightarrow{x}^{(0)}\right)$
3: **while** Norm$[\overrightarrow{x}^{(n)} - \overrightarrow{x}^{(n-1)}] \geq \varepsilon$ **do**
4: $\quad x_{\text{pom}} = \overrightarrow{x}^{(n)}$
5: $\quad \overrightarrow{x}^{(n+1)} = x_{\text{pom}} - J^{-1}(x_{\text{pom}})\overrightarrow{f}\,(x_{\text{pom}})$
6: $\quad \overrightarrow{x}^{(n)} = x_{\text{pom}}$
7: $i := i + 1$
8: **end while**
9: **return** $\overrightarrow{x}^{(n)}$

Teorema 18. ***(Kantorovič)*** *Neka je $f_i \in \mathcal{C}^{(2)}(D), i \in \{1, ..., n\}$ gde je $D \subset \mathbb{R}^n$ oblast u kojoj jednačina*

$$\overrightarrow{f}(\overrightarrow{x}) = \overrightarrow{0}$$

ima jednistveno rešenje $\overrightarrow{\overline{x}} = [\overline{x_1}\ \overline{x_2} \cdots \overline{x_n}]^T$. Ukoliko je $\det J\left(\overrightarrow{\overline{x}}\right) \neq 0$, tada postoji okolina $U \subset D$ tačke $\overrightarrow{\overline{x}} = [\overline{x_1}\ \overline{x_2} \cdots \overline{x_n}]^T$, takva da je metod Njutna-Kantoroviča konvergentan za proizvoljnu startnu vrednost $\overrightarrow{x}^{(0)}$ iz te okoline. U tom slučaju je red konvegencije metoda jednak $r = 2$.

Dokaz ove teoreme se može naći u [2].

3.1.1 Implementacija

Implementaciju dajemo u dvodimenzionalnom slučaju, radi geometrijske preglednosti.

```
Jacobi[f1_, f2_, tacka_] :=
  Module[{a = tacka[[1, 1]], b = tacka[[2, 1]], W, V,
    x = Variables[f1][[1]], y = Variables[f1][[2]]},
   V = {{D[f1, x], D[f1, y]}, {D[f2, x], D[f2, y]}};
   W = V /. x -> a /. y -> b;
   Return[W]]

NjutnKantorovic[f1_, f2_, eps_, xstart_] :=
 Module[{Tabelica = {}, xnew, xold, xpom, i, x = Variables[f1][[1]],
   y = Variables[f1][[2]]},
  Tabelica = {{"i", "X[i]"}};
  xold = xstart;
  i = 0;
  AppendTo[Tabelica, {i, MatrixForm[xold]}];
  xnew = xold -
    Inverse[Jacobi[f1, f2,
       xold]].{{f1 /. x -> xold[[1, 1]] /.
        y -> xold[[2, 1]]}, {f2 /. x -> xold[[1, 1]] /.
        y -> xold[[2, 1]]}};
  i++;
```

```
While[Norm[xnew - xold, "Frobenius"] >= eps,
 xpom = xnew;
 xnew =
  xpom - Inverse[
     Jacobi[f1, f2,
      xpom]].{{f1 /. x -> xpom[[1, 1]] /.
       y -> xpom[[2, 1]]}, {f2 /. x -> xpom[[1, 1]] /.
       y -> xpom[[2, 1]]}};
 AppendTo[Tabelica, {i, MatrixForm[N[xnew, 15]]}];
 xold = xpom;
 i++
 ];
Return[TableForm[Tabelica]]
]
```

Primer 12. *Sa tačnošću* $\varepsilon = 10^{-6}$ *naći jedno rešenje sistema*

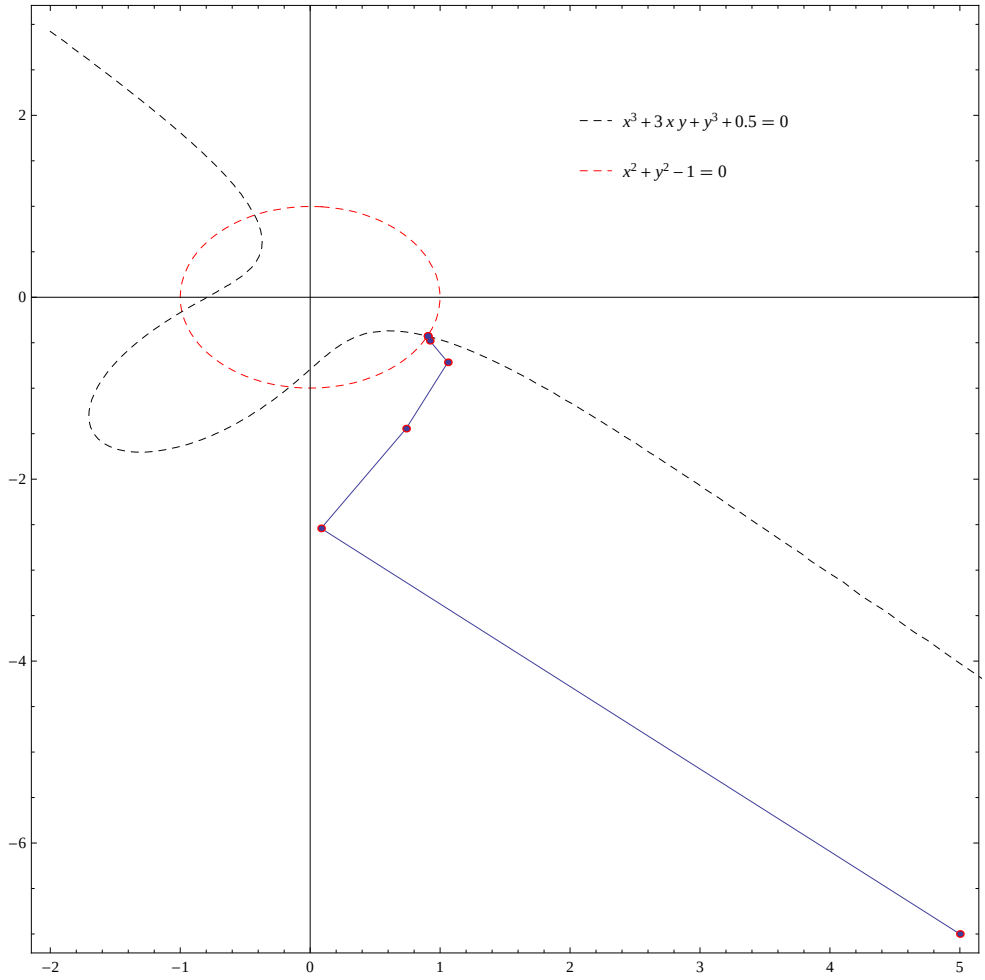
$$(\mathcal{S}) : \begin{cases} x^2 + y^2 - 1 = 0 \\ x^3 + y^3 + 3xy + \frac{1}{2} = 0. \end{cases}$$

metodom Njutna-Kantoroviča.

Rešenje. Startujući od početne aproksimacije $[x_0, y_0]^T = [7, -4]^T$ dobijamo sledeće vrednosti

$$\left\{ \begin{array}{cc} \text{i} & \text{X[i]} \\ 0 & \begin{pmatrix} 5 \\ -7 \end{pmatrix} \\ 1 & \begin{pmatrix} 0.0841747 \\ -2.54081 \end{pmatrix} \\ 2 & \begin{pmatrix} 0.738395 \\ -1.44412 \end{pmatrix} \\ 3 & \begin{pmatrix} 1.05948 \\ -0.715343 \end{pmatrix} \\ 4 & \begin{pmatrix} 0.920618 \\ -0.477715 \end{pmatrix} \\ 5 & \begin{pmatrix} 0.904478 \\ -0.429536 \end{pmatrix} \\ 6 & \begin{pmatrix} 0.903858 \\ -0.427836 \end{pmatrix} \\ 7 & \begin{pmatrix} 0.903857 \\ -0.427834 \end{pmatrix} \\ 8 & \begin{pmatrix} 0.903857 \\ -0.427834 \end{pmatrix} \end{array} \right\}$$

Kako je sistem simetričan u odnosu na koordinatni početak, po nepoznatim x i y, možemo zaključiti da je još jedno rešenje sistema $[x, y]^T = [-0.903857, +0.427834]^T$.



Primetimo takođe da je ovde moguće primeniti mogućnost **simboličkog računanja**. Naime, pozivom funckije dobijamo `Jacobi`[f1,f2, {{x}, {y}}] dobijamo Jakobijevu matricu

$$J(x,y) = \begin{bmatrix} 2x & 2y \\ 3x^2+3y & 3y^2+3x \end{bmatrix}.$$

Pozivom funkcije `Inverse` dobijamo matricu inverznu Jakobijevoj matrici u tački (x,y)

$$J^{-1}(x,y) = \begin{bmatrix} \frac{3y^2+3x}{-6yx^2+6x^2+6y^2x-6y^2} & -\frac{2y}{-6yx^2+6x^2+6y^2x-6y^2} \\ \frac{-3x^2-3y}{-6yx^2+6x^2+6y^2x-6y^2} & \frac{2x}{-6yx^2+6x^2+6y^2x-6y^2} \end{bmatrix}.$$

Sada ne treba u svakoj iteraciji algoritma da izračunavamo inverznu matricu, nego samo da menjamo x i y odgovarajućim vrednostima. △

3.2 Gradijentni metod

Prezentovaćemo još jedan metod za rešavanje sistema nelinarnih jednačina. Pre toga su nam potrebni neki osnovni pojmovi i tvrđenja.

Definicija 6. *Neka je data funkcija* $f : \mathbb{R}^n \to \mathbb{R}$. *Vektor* $\nabla f(x_1,...,x_n) = \left[\frac{\partial f}{\partial x_1},...,\frac{\partial f}{\partial x_n}\right]^T$ *naziva se* ***gradijent*** *funkcije* f *u tački* $(x_1,...,x_n) \in \mathbb{R}^n$ *(ako postoji).*

Definicija 7. ***Skalarno polje*** *je svaka funkcija* $U : \mathbb{R}^n \to \mathbb{R}$.

Teorema 19. *Neka je $f : \mathbb{R}^2 \to \mathbb{R}$ skalarna funkcija koja ima neprekidne prve parcijalne izvode. Tada gradijent funkcije f postoji i nezavisan je od izbora koordinatnog sistema u prostoru $\mathbb{R}^2$.*

*U svakoj tački u kojoj je gradijent funkcije različit od $\overrightarrow{0} = (0,0)$ važi da gradijent funkcije predstavlja smer **najbržeg rasta** skalarnog polja određenog funkcijom f.*

Dokaz. Činjenica da gradijent funkcije f postoji i da je nezavisan od izbora odabira koordinatnog početka sledi direktno iz uslova teoreme.

Neka je $(a_x, a_y) = \overrightarrow{a} \neq \overrightarrow{0}$ proizvoljan pravac. Kako su parcijalni izvodi neprekidne funkcije imamo da važi

$$f'_{\overrightarrow{a}}(x,y) = \frac{\partial f(x,y)}{\partial x_1} a_x + \frac{\partial f(x,y,z)}{\partial y} a_y = \nabla f(x,y) \cdot (a_x, a_y).$$

Sa druge strane, po definiciji skalarnog proizvoda, imamo

$$\nabla f(x,y) \cdot (a_x, a_y) = ||\nabla f(x,y)|| \cdot ||\overrightarrow{a}|| \cdot \cos \angle(\nabla f(x,y), \overrightarrow{a}).$$

Odavde vidimo da je $f'_{\overrightarrow{a}}(x,y)$ najveće kada su vektori $\overrightarrow{a}$ i $\nabla f(x,y)$ kolinearni. Kako parcijalni izvod u pravcu nekog vektora "meri"rast funkcije u ravni koju odredjuje taj vektor i njegova normalna projekcija na xOy ravan, imamo tvrđenje teoreme. □

Posledica 3. *Neka je $f : \mathbb{R}^2 \to \mathbb{R}$ skalarna funkcija koja ima neprekidne prve parcijalne izvode. U svakoj tački u kojoj je gradijent funkcije različit od $\overrightarrow{0} = (0,0)$ važi da vektor koji je **suprotan** gradijentu funkcije predstavlja smer **najbržeg opadanja** skalarnog polja određenog funkcijom f.*

Teorema 20. *Neka je $f : \mathbb{R}^2 \to \mathbb{R}$ funkcija koja ima neprekidne prve parcijalne izvode. Neka je (x_0, y_0) proizvoljna tačka ortogonalne projekcije tačaka krive na xOy ravan za koju važi da je $f(x,y) = c$. Tada je gradijentni vektor $\nabla f(x_0, y_0)$ normalan na tu ortogonalnu projekciju tačaka.*

Dokaz. Zapišimo dati skup tačaka $f(x,y) = c$ u parametarskom obliku

$$f(x(t), y(t)) = c, \ t \in [a,b].$$

Koristeći teoremu o izvodu složene funkcije dobijamo

$$\frac{\partial f(x(t),y(t))}{\partial x} x'(t) + \frac{\partial f(x(t),y(t))}{\partial y} y'(t) = 0, \quad t \in [a,b].$$

Na osnovu prethodne jednakosti imamo

$$\left(\frac{\partial f(x(t),y(t))}{\partial x}, \frac{\partial f(x(t),y(t))}{\partial y} \right) \cdot (x'(t), y'(t)) = 0, \quad t \in [a,b].$$

Specijalno, imamo da je

$$\left(\frac{\partial f(x(t_0),y(t_0))}{\partial x}, \frac{\partial f(x(t_0),y(t_0))}{\partial y} \right) \cdot (x'(t_0), y'(t_0)) = 0,$$

gde je $t_0 \in [a,b]$ za koje je $x(t_0) = x_0$ i $y(t_0) = y_0$. Na osnovu ove jednakosti sledi tvrđenje teoreme. □

Prethodne dve teoreme se direktno uopštavaju na višedimenzionalni slučaj, ali zbog jednostavnosti i geometrijske interpretacije date se u dvodimenzionalnom slučaju.

3.2.1 Algoritam najbržeg pada

Neka je dat nelinearni sistem

$$(\mathcal{S}_n): \begin{cases} f_1(x_1, x_2, ..., x_n) = 0 \\ f_2(x_1, x_2, ..., x_n) = 0 \\ \qquad ... \\ f_n(x_1, x_2, ..., x_n) = 0. \end{cases}$$

Ovaj sistem je ekvivalentan jednačini

$$\sum_{i=1}^{n} (f_i(x_1, x_2, ..., x_n))^2 = 0.$$

Definišimo **funkcionelu** $U : \mathbb{R}^n \to \mathbb{R}$ sa

$$U(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} (f_i(x_1, x_2, ..., x_n))^2.$$

Cilj je **minimizovati** funkcionelu U. Neka je $\overrightarrow{x} = \overrightarrow{a}$ jedinstveno rešenje za koje funkcionela U dostiše minimum i neka je $\overrightarrow{x}^{(0)}$ početna aproksimacija. Konstruišimo niz $\left(\overrightarrow{x}^{(k)}\right)_{k=0}^{+\infty}$ takav da je $U\left(\overrightarrow{x}^{(0)}\right) > U\left(\overrightarrow{x}^{(1)}\right) > U\left(\overrightarrow{x}^{(2)}\right) > U\left(\overrightarrow{x}^{(3)}\right) > ...$.

Neka je

$$\overrightarrow{x}^{(k+1)} = \overrightarrow{x}^{(k)} - \lambda_k \nabla U\left(\overrightarrow{x}^{(k)}\right), \quad k \in \mathbb{N} \cup \{0\}.$$

Prethodna jednakost je logična jer ako je $\overrightarrow{x}^{(k)}$ tekuća iteracija, do sledeće dolazimo tako što se krećemo **suprotno** od gradijenta skalarnog polja U a dužinu koraka određujemo u svakoj iteraciji sa parametrom λ_k.

Parametar λ_k određujemo iz uslova da skalarna funkcija $S : \mathbb{R} \to \mathbb{R}$ definisana sa

$$S(t) = U\left(\overrightarrow{x}^{(k)} - tU\left(\overrightarrow{x}^{(k)}\right)\right)$$

imam minimum u tački $t = \lambda_k$. Linearizacijom jednačine $S'(t) = 0$ dobijamo da je

$$\lambda_k = \frac{\sum_{i=1}^{n} \left(\nabla f_i\left(\overrightarrow{x}^{(k)}\right), \nabla U\left(\overrightarrow{x}^{(k)}\right) f_i\left(\overrightarrow{x}^{(k)}\right)\right)}{\sum_{i=1}^{n} \left(\nabla f_i(\overrightarrow{x}^{(k)}), \nabla U\left(\overrightarrow{x}^{(k)}\right)\right)^2}, \quad k \in \mathbb{N} \cup \{0\}.$$

Kako je

$$\frac{\partial U}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\sum_{i=1}^{n} (f_i(x_1, x_2, ..., x_n))^2\right) = 2\sum_{i=1}^{n} f_i(x_1, x_2, ..., x_n) \frac{\partial f_i(x_1, x_2, ..., x_n)}{\partial x_j}$$

dobijamo da je

$$\nabla U\left((x_1, x_2, ..., x_n)\right) = 2J^T((x_1, x_2, ..., x_n))\overrightarrow{f}((x_1, x_2, ..., x_n)).$$

Sada je

$$\lambda_k = \frac{1}{2} \frac{\left(\overrightarrow{f}^{(k)}, J_k J_k^T \overrightarrow{f}^{(k)}\right)}{\left(J_k J_k^T \overrightarrow{f}^{(k)}, J_k J_k^T \overrightarrow{f}^{(k)}\right)},$$

gde je $\overrightarrow{f_k} = \overrightarrow{f}\left(\overrightarrow{x}^{(k)}\right)$ i $J_k = J\left(\overrightarrow{x}^{(k)}\right)$. Sada iterativni metod poprima oblik

$$\overrightarrow{x}^{(k+1)} = \overrightarrow{x}^{(k)} - 2\lambda_k J_k^T \overrightarrow{f_k}, \quad k \in \mathbb{N} \cup \{0\}.$$

Dajemo pseudokod ovog algoritma.

Zaustavni kriterijum će biti, kao i kod metoda Njutna-Kantoroviča, Frobenijusova norma razlike susednih iteracija.

Algoritam 12 Gradijentni metod

Input Sistem nelinarnih jednačina, početni vektor $\overrightarrow{x}^{(0)}$ i tačnost $\varepsilon > 0$.
1: $i := 1$
2: $\overrightarrow{x}^{(1)} = \overrightarrow{x}^{(0)} - 2\lambda_0 (J(\overrightarrow{x}^{(0)}))^T \overrightarrow{f}\left(\overrightarrow{x}^{(0)}\right)$
3: **while** Norm$\left[\overrightarrow{x}^{(n)} - \overrightarrow{x}^{(n-1)}\right] \geq \varepsilon$ **do**
4: $\quad \overrightarrow{x_{\text{pom}}} = \overrightarrow{x}^{(n)}$
5: $\quad \overrightarrow{x}^{(n+1)} = x_{\text{pom}} - 2\lambda_n (J(\overrightarrow{x_{\text{pom}}}))^T \overrightarrow{f}(\overrightarrow{x_{\text{pom}}})$
6: $\quad \overrightarrow{x}^{(n)} = \overrightarrow{x_{\text{pom}}}$
7: $i := i + 1$
8: **end while**
9: **return** $\overrightarrow{x}^{(n)}$

Ovaj metod ima **linearnu konvergenciju**. Iterativni koraci su na početku najveći, dok kako se približavamo rešenju su sve sitniji, dok na kraju osciluju, praveći "cik-cak" korake oko rešenja. Zato se u praksi koristi ovaj metod na početku, pa kada se približimo rešenju, nastavlja se sa nekim bržim metodom (na primer Njutn-Kantorovičev).

Za više informacija videti [4].

3.2.2 Implementacija

Primer 13. *Sa tačnošću* $\varepsilon = 10^{-6}$ *naći jedno rešenje sistema*

$$(\mathcal{S}) : \begin{cases} x^2 + y^2 - 1 = 0 \\ x^3 + y^3 + 3xy + \frac{1}{2} = 0 \end{cases}$$

gradijentnim metodom.

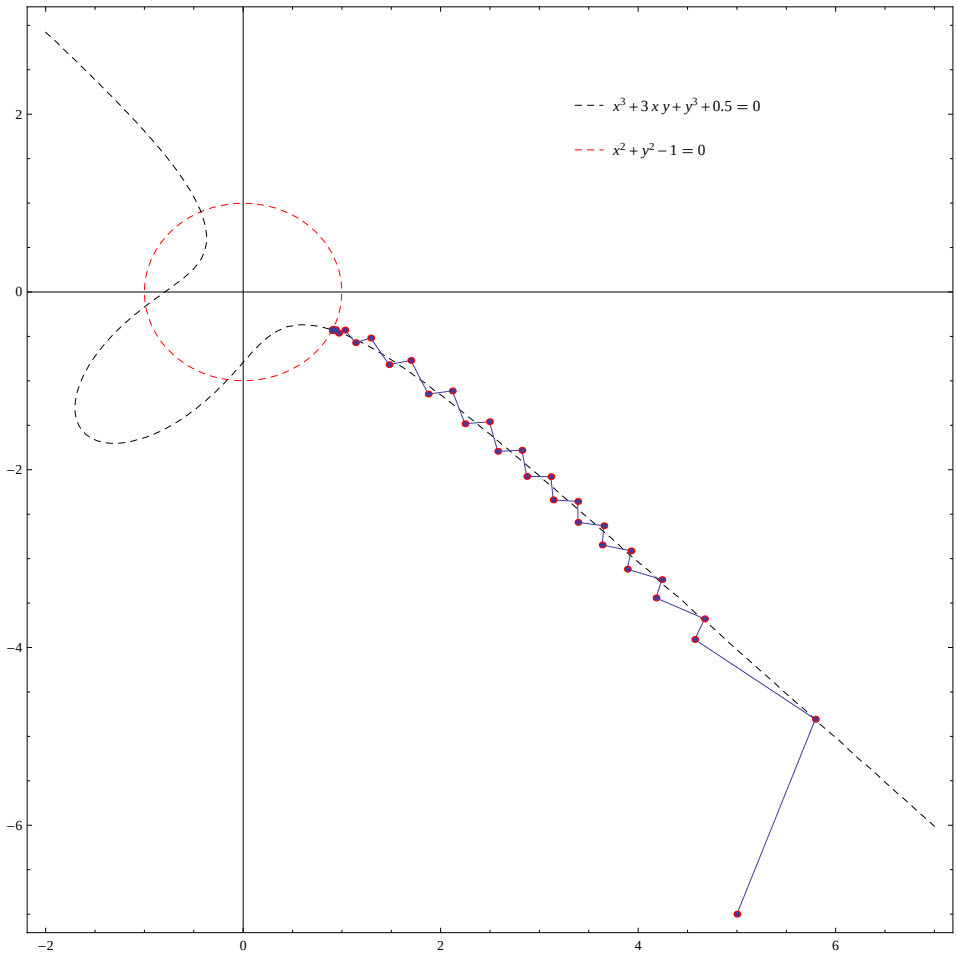
Rešenje. Startujući od početne aproksimacije $[x_0, y_0]^T = [7, -4]^T$, dobijamo posle 42 iteracije rešenje $[x, y]^T = [0.903858 \ -0.427834]^T$. Iteracije ne navodimo zbog uštede prostora.

```
Jacobi[f1_, f2_, tacka_] :=
 Module[{a = tacka[[1, 1]], b = tacka[[2, 1]], W, V,
   x = Variables[f1][[1]], y = Variables[f1][[2]]},
  V = {{D[f1, x], D[f1, y]}, {D[f2, x], D[f2, y]}};
  W = V /. x -> a /. y -> b;
  Return[W]]

Lambda[f1_, f2_, tacka_] := Module[
```

```
  {rez, F, vek, vektor, vrednost1, vrednost2, x, y},
  {x, y} = Variables[f1];
  vrednost1 = f1 /. x -> tacka[[1, 1]] /. y -> tacka[[2, 1]];
  vrednost2 = f2 /. x -> tacka[[1, 1]] /. y -> tacka[[2, 1]];
  F = {vrednost1, vrednost2};
  vek =
   Jacobi[f1, f2, tacka].Transpose[
     Jacobi[f1, f2, tacka]].{{vrednost1}, {vrednost2}};
  vektor = {vek[[1, 1]], vek[[2, 1]]};
  rez = (1/2)*(F.vektor)/(vektor.vektor);
  Return[rez]
  ]

GradijentniMetod[f1_, f2_, eps_, xstart_] :=
 Module[{Tabelica = {}, ListaTacaka = {}, xnew, xold, xpom, i,
   x = Variables[f1][[1]], y = Variables[f1][[2]]},
  Tabelica = {{"i", "X[i]"}};
  xold = xstart;
  i = 0;
  AppendTo[Tabelica, {i, MatrixForm[xold]}];
  xnew = xold -
    2*Lambda[f1, f2, xold]*
     Transpose[
       Jacobi[f1, f2,
        xold]].{{f1 /. x -> xold[[1, 1]] /.
         y -> xold[[2, 1]]}, {f2 /. x -> xold[[1, 1]] /.
         y -> xold[[2, 1]]}};
  i++;
  While[Norm[xnew - xold, "Frobenius"] >= eps,
   xpom = xnew;
   xnew =
    xpom - 2*Lambda[f1, f2, xpom]*
      Transpose[
        Jacobi[f1, f2,
         xpom]].{{f1 /. x -> xpom[[1, 1]] /.
          y -> xpom[[2, 1]]}, {f2 /. x -> xpom[[1, 1]] /.
          y -> xpom[[2, 1]]}};
   AppendTo[Tabelica, {i, MatrixForm[xnew]}];
   xold = xpom;
   i++
   ];
  Return[TableForm[Tabelica]]
  ]
```



Sa slike možemo uočiti da je veličina koraka na početku iterativnog procesa najveća a da kako napredujemo ka rešenju sve se više smanjuje praveći "cik-cak" korake oko rešenja. △

4 Zaključak

Kroz ovaj rad smo videli veliki značaj numeričkih i simboličkih mogućnosti programskog paketa `Mathematica` u Numeričkoj matematici.

Praktično nijedno istraživanje nije moguće izvršiti bez upotrebe nekog dobrog softvera, jer je nezamislivo da čovek "ručno" izračuna sve korake i najprostijeg algoritma. Čak i da je to moguće posao bi bio vrlo zamoran a najverovatnije će doći i do neke greške uslovljene ljudskim faktorom.

Kako se nelinearne jednačine sreću u raznim naukama koje nisu obavezno matematičke, kao što je ekonomija, biologija ili demografija, važno je imati dobar algoritam koji će ih rešavati.

Aritmetika proizvoljne preciznosti nam omogućava da pristupimo rešavanju problema sa onolikom tačnošću koja će zadovoljiti praktične potrebe.

Pored numeričkih i simboličkih mogućnosti koje nam pruža programski paket `Mathematica`, izdvojimo i izvanredne grafičke mogućnosti. Sve slike i grafici iz ovog rada su načinjeni u programskom paketu `Mathematica`.

Imajući na umu grafiku, programski paket `Mathematica`, može biti ne samo naučno sredstvo, nego i izvanredna pomoć kod priprema predavanja u osnovnim,

srednjim školama i na visokoškolskim ustanovama.

5 Zahvalnost

Želim ovom prilikom da se posebno zahvalim profesoru dr Predragu Stanimiroviću na pomoći oko izrade ovog rada kao i profesoru dr Nebojši Dinčiću koji je pročitao rad i dao niz korisnih sugestija. Takođe, želim da se zahvalim i profesoru dr Predragu Krtolici koji je ukazao na tehničke nedostatke teksta.

Literatura

[1] Marko Petković, *Algoritmi Numeričke analize*, Prirodno-matematički fakultet, Niš, 2013.

[2] Gradimir Milovanović, *Numerička anliza I*, Naučna knjiga, Beograd, 1988.

[3] Predrag Stanimirovović, Gradimir Milovanović, *Programski paket `Mathematica` i primene*, Elektronski fakultet, Niš, 2002.

[4] Predrag Stanimirović, Marko Miladinović, *Nelinearna optimizacija*, Prirodno-matematički fakultet, Niš, 2014.

[5] Gradimir Milovanović, Milan Kovačević, Miodrag Spalević, *Numerička matematika-zbirka rešenih problema*, Elektronski fakultet, Univerzitet u Nišu, 2003.

[6] Ljiljana Petković, Slobodan Tričković, Predrag Rajković, *Zbirka zadataka iz Numeričke matematike*, Nova Jugoslavija, Vranje, 1997.

[7] Desanka Radunović, Aleksandar Samardžić, Filip Marić, *Numeričke metode-Zbirka zadataka kroz `C`, `Fortran` i `MatLab`*, Akademska misao, Beograd, 2005.

[8] Snežana Živković Zlatanović, Marko Đikić, *Matematička analiza I*, Prirodno-matematički fakultet, Niš, 2014.

[9] Radoslav Dimitrijević, *Analiza realnih funkcija više promenljivih*, Prirodno-matematički fakultet, Niš, 2014.

[10] Radoslav Dimitrijević, *Zbirka zadataka iz teorije polinoma*, Prirodno-matematički fakultet, Niš, 2007.

[11] Thomas Cormen, Charles Leiserson, Ronald Rivest, Clifford Stein, *Introduction to algorithms*, Mit Press, 2009.

[12] Gradimir Milovanović, *Numerička analiza i teorija aproksimacija*, Naučna knjiga, Beograd, 2014.

[13] Desanka Radunović, *Numerička matematika, Udžbenik za 4. razred Matematičke gimnazije*, Krug, Beograd, 2011.

[14] Mimica Milošević, *Iterativni metodi za aproksimaciju nula polinoma*, Univerzitet u Nišu, Prirodno-matematički fakultet, 2011.

[15] Jovana Džunić, *Višekoračni metodi za rešavanje nelinearnih jednačina*, Univerzitet u Nišu, Prirodno-matematički fakultet, 2012.

[16] Žikica Perović, *Algebra I*, Prirodno-matematički fakultet, Niš, 2002.

[17] Vladimir Rakočević, *Funkcionalna analiza*, Naučna knjiga, Beograd, 1994.