Metode linijskog pretraživanja

Trošić, Ivana

Master's thesis / Diplomski rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Department of Mathematics / Sveučilište Josipa Jurja Strossmayera u Osijeku, Odjel za matematiku**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:126:158155

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni diplomski studij
matematike i računarstva

Ivana Trošić

Metode linijskog pretraživanja

Osijek, 2017.

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni diplomski studij
matematike i računarstva

Ivana Trošić

Metode linijskog pretraživanja

Diplomski rad

Mentor: prof. dr. sc. K. Sabo

Osijek, 2017.

Sadržaj

1 Uvod

Linijsko pretraživanje je jedna od lokalnih iterativnih metoda bezuvjetne višedimenzionalne optimizacije. Promotrimo kako općenito izgleda problem višedimenzionalne minimizacije. Općenito, polazimo od problema traženja točke u kojoj se postiže globalni minimum funkcije $f : \mathbb{R}^n \to \mathbb{R}$, tj. tražimo $x^* \in \mathbb{R}^n$ takav da je $x^* = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} f(x)$. Umjesto traženja točke globalnog minimuma, u primjenama nam je često dovoljno odrediti točku u kojoj se postiže lokalni minimum funkcije. Osnova metode linijskog pretraživanja sastoji se u sljedećem: za danu $k-$tu aproksimaciju $x_k \in \mathbb{R}^n$, sljedeću $(k+1)-$vu aproksimaciju x_{k+1} tražimo u obliku:

$$x_{k+1} = x_k + \alpha_k p_k \qquad k = 0, 1, 2, \ldots$$

Pritom je x_0 početna aproksimacija, $p_k \in \mathbb{R}^n$ vektor smjera, a $\alpha_k > 0$ duljina koraka. Pomak iz aproksimacije x_k u aproksimaciju x_{k+1} treba ostvariti na način da se smanjuje vrijednost funkcije tj. da vrijedi

$$f(x_{k+1}) = f(x_k + \alpha_k p_k) < f(x_k). \tag{1}$$

Cilj nam je pronaći odgovarajući vektor smjera $p_k \in \mathbb{R}^n$ te duljinu koraka $\alpha_k > 0$. Ako pretpostavimo da je poznat vektor smjera p_k, onda se problem pronalaska α_k svodi na linijsko pretraživanje, odnosno jednodimenzionalnu minimizaciju po varijabli $\alpha > 0$. Neka je

$$\phi(\alpha) = f(x_k + \alpha p_k). \tag{2}$$

Želimo pronaći duljinu koraka $\alpha_k > 0$ u smjeru vektora smjera p_k tako da je

$$\phi(\alpha_k) < \phi(0). \tag{3}$$

Iskoristimo li jednakost (2) dobivamo

$$f(x_{k+1}) = f(x_k + \alpha_k p_k) < f(x_k + 0 \cdot p_k) = f(x_k).$$

Uočimo da zadovoljavanjem uvjeta (3) ne narušavamo početni uvjet (1). Pretpostavimo da je $\alpha_k > 0$ takav da je funkcija f u smjeru p_k minimalna tj.

$$f(x_k + \alpha_k p_k) = \min_{\alpha > 0} f(x_k + \alpha p_k). \tag{4}$$

Odnosno govoreći u terminima funkcije ϕ imamo

$$\phi(\alpha_k) = \min_{\alpha > 0} \phi(\alpha). \tag{5}$$

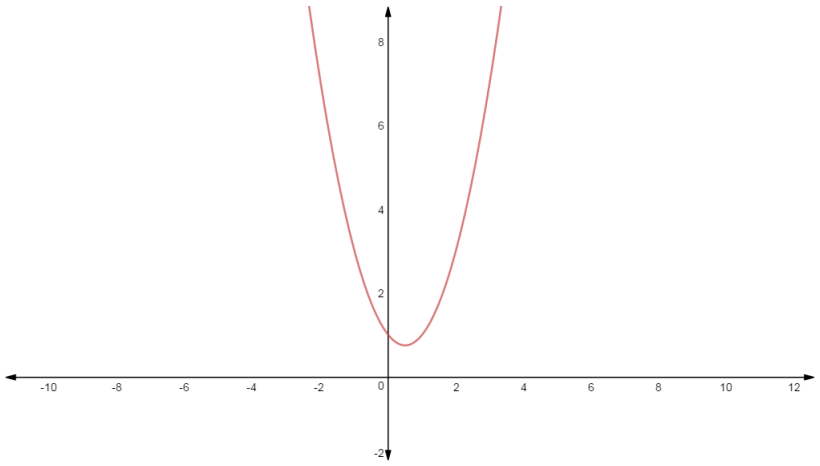
Ako vrijedi (4), onda govorimo o egzaktnom ili optimalnom linijskom pretraživanju, a $\alpha_k > 0$ zovemo optimalna duljina koraka. Ako pak zadovoljimo uvjet

$$f(x_{k+1}) - f(x_k) = f(x_k + \alpha_k p_k) - f(x_k) < 0,$$

govorimo o aproksimativnom ili približnom linijskom pretraživanju. Kod praktičnih primjera, puno češće se upotrebljava aproksimativno linijsko pretraživanje, jer je optimalno nemoguće postići ili je računski puno zahtjevnije. Također, ako imamo puno iteracija egzaktno linijsko pretraživanje nije učinkovito. Kod egzaktnih metoda linijskog pretraživanja predstavljenih u ovom radu zahtjevat ćemo da funckija ϕ bude unimodalna. Sljedeća definicija govori nam upravo o tom svojstvu.

Definicija 1 *Kažemo da je funkcija* $\phi : [a,b] \to \mathbb{R}$ *strogo unimodalna na intervalu* $[a,b]$ *ako postoji* $x^* \in [a,b]$ *takav da je* $\phi\big|_{[a,x^*]}$ *strogo monotono padajuća te da je* $\phi\big|_{[x^*,b]}$ *strogo monotono rastuća.*

Primjer 1 *Neka je* $\phi(\alpha) = \alpha^2 - \alpha + 1$, $\alpha \in [-2,2]$ *(vidi Sliku 1). Provjerimo da li je funkcija* ϕ *unimodalna.*



Slika 1: Grafički prikaz funkcije $\phi(\alpha) = \alpha^2 - \alpha + 1$

Kako se radi o kvadratnoj funkciji, graf funkcije je parabola što možemo vidjeti i na Slici 1. Lako se može izračunati da je tjeme funkcije točka $T_1 = (\frac{1}{2}, \frac{3}{4})$. Kako je koeficijent ispred kvadratnog člana pozitivan, parabola je okrenuta "prema gore", te vrijedi sljedeće:

- za $\alpha \in [-2, \frac{1}{2}]$ funkcija ϕ je monotono padajuća
- za $\alpha \in [\frac{1}{2}, 2]$ funkcija ϕ je monotono rastuća

Prema Definiciji 1 slijedi da je ϕ unimodalna.

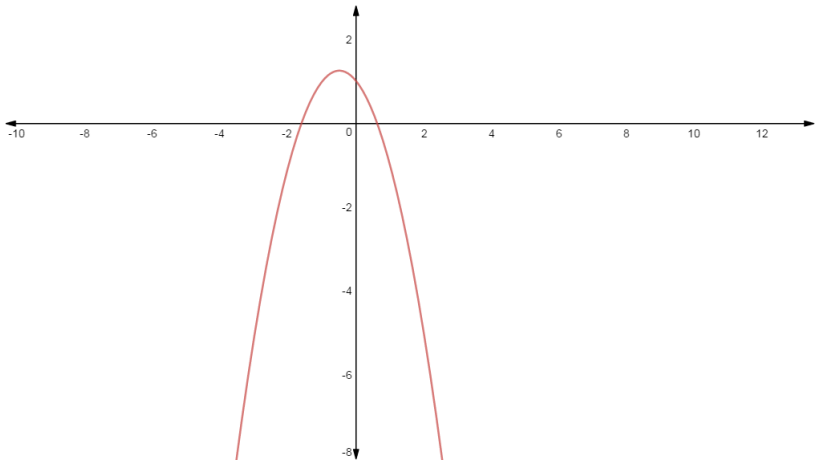
Primjer 2 *Neka je* $\phi(\alpha) = -\alpha^2 - \alpha + 1$, $\alpha \in [-2,2]$ *(vidi Sliku 2). Provjerimo da li je funkcija* ϕ *unimodalna.*

U ovome primjeru također imamo kvadratnu funkciju ali s negativnim koeficijentom uz kvadratni član. Može se izračunati da je tjeme funkcije $T_2 = (-\frac{1}{2}, \frac{3}{2})$. Kao što možemo vidjeti sa Slike 2 parabola je okrenuta "prema dolje" te vrijedi sljedeće:

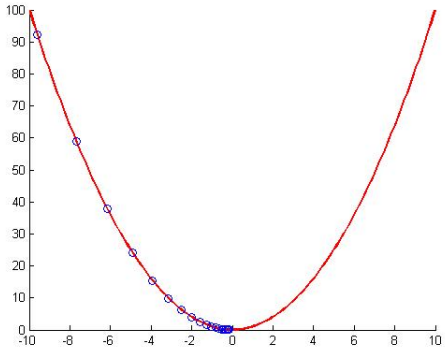
- za $\alpha \in [-2, -\frac{1}{2}]$ funkcija ϕ je monotono rastuća
- za $\alpha \in [-\frac{1}{2}, \frac{3}{2}]$ funkcija ϕ je monotono padajuća

Kako ne vrijedi Definicija 1, tj. ne postoji takav $\alpha \in [-2,2]$ koji bi zadovoljavao uvjete iz definicije, slijedi da ϕ nije unimodalna.

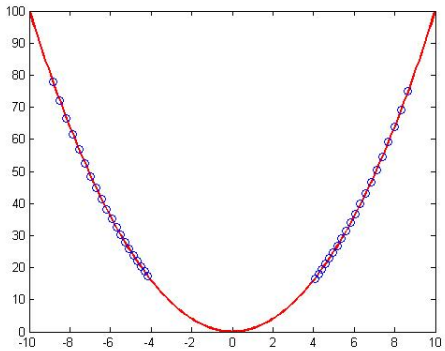
Izbor α_k unutar svake pojedine iteracije izuzetno je bitan kako bi metoda konvergirala, te kako bi u što kraćem broju koraka došli do optimalnog, odnosno aproksimativnog rješenja optimizacijskog problema. Možemo vidjeti sa Slike 3a da ako je duljina koraka α premala



Slika 2: Grafički prikaz funkcije $\phi(\alpha) = -\alpha^2 - \alpha + 1$



(a) Duljina koraka $\alpha = 0.1$



(b) Duljina koraka $\alpha = 2$

Slika 3: Usporedni prikaz gradijentne metode s različitim konstatnim duljinama koraka α

uvelike povećavamo broj iteracija metode, odnosno metoda konvergira sporo. S druge strane ako je duljina koraka α prevelika, kao što možemo vidjeti na Slici 3b, metoda divergira.

Cilj ovog rada je razmotriti i analizirati različite metode za određivanje duljine koraka $\alpha_k > 0$. Prije toga u sljedećoj točki navodimo jednu metodu za određivanje vektora smjera p_k.

2 Određivanje vektora smjera

Kao što je već spomenuto u Poglavlju 1 želimo da pomak iz aproksimacije x_k u aproksimaciju x_{k+1} bude ostvaren na način da se smanjuje vrijednost funkcije tj. da vrijedi $f(x_{k+1}) = f(x_k + \alpha_k p_k) < f(x_k)$. To se može postići tako da za vektor smjera $p_k \in \mathbb{R}^n$ vrijedi

$$\nabla f(x_k)^T p_k < 0$$

pri čemu je $\nabla f(x_k)$ gradijent funkcije f u točki x_k. Iskažimo teorem koji nam govori o tome.

Teorem 1 ([8]) *Neka je funkcional $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ diferencijabilna u svakoj točki $x \in Int(D)$ i neka je $\nabla f(x)^T p < 0$ za neki $p \in \mathbb{R}^n$. Onda postoji $\delta > 0$, takav da*

$$f(x + \alpha p) < f(x), \forall \alpha \in (0, \delta).$$

Treba odabrati vektor koraka p_k takav da zadovoljava Teorem 1. Kroz sljedeće podpoglavlje pokazat ćemo jednu takvu metodu.

2.1 Gradijentna metoda

Znamo da je $\phi(\alpha) = f(x_k + \alpha p_k)$, stoga je

$$\phi'(\alpha) = (\nabla f(x_k + \alpha p_k))^T p_k.$$

Odnosno, $\phi'(0) = (\nabla f(x_k))^T p_k$. Neka je p_k jedinični vektor, tj. neka je $||p_k|| = 1$, iz definicije skalarnog produkta slijedi

$$\phi'(0) = ||\nabla f(x_k)|| \cos\theta$$

pri čemu je θ kut između vektora $\nabla f(x_k)$ i p_k. Nadalje, $\phi'(0)$ je minimalna za $\theta = \pi$ iz čega slijedi da je

$$p_k = -\frac{\nabla f(x_k)}{||\nabla f(x_k)||} \tag{6}$$

$$\phi'(0) = -||\nabla f(x_k)||. \tag{7}$$

Stoga u iterativnom postupku minimizacije funkcije $f : \mathbb{R}^n \to \mathbb{R}$, pri čemu je $\alpha_k > 0$ duljina koraka,

$$x_{k+1} = x_k + \alpha_k p_k \qquad k = 0, 1, 2, \ldots$$

za p_k uzimamo smjer negativnog gradijenta, točnije $p_k = -\nabla f(x_k)$. Upravo to je gradijentna metoda. Uočimo, ovakav izbor vektora smjera p_k zadovoljava Teorem 1

$$\nabla f(x_k)^T p_k = -\nabla f(x_k)^T \nabla f(x_k) = -||\nabla f(x_k)|| < 0.$$

Zaključujemo da je vektor smjera $p_k = -\nabla f(x_k)$ dobar izbor jer nam u svakom koraku osigurava smanjenje funkcije $f : \mathbb{R}^n \to \mathbb{R}$.

3 Određivanje početnog intervala koji sadrži duljinu koraka

Prije primjene same metode izabiremo početni interval $I = [a, b]$ koji će sadržavati $\alpha_k > 0$ takav da zadovoljava uvjet (5), označimo ga sa α^*. Preciznije, izabiremo zatvoreni interval $[a, b] \subset [0, +\infty)$ takav da je $\alpha^* \in [a, b]$. Ideja je da zatim početni interval metodama linijskog pretraživanja smanjujemo, pazeći naravno da vrijedi $\alpha^* \in I$. Predstaviti ćemo "naprijed-nazad" metodu kojom ćemo odrediti početni interval koristeći svojstvo unimodalnosti funkcije ϕ. Ideja metode je da za početnu točku $\alpha_0 > 0$, te za dani $h_0 > 0$ provjerimo koje vrijednosti postiže funkcija ϕ, te na temelju tih vrijednosti odlučimo u kojem ćemo se "smjeru" pomaknuti. Primijetimo sljedeće, ako je:

$$\phi(\alpha_0 + h_0) < \phi(\alpha_0), \tag{8}$$

onda se pomičemo u "dobrom smjeru", tj. smanjujemo vrijednost funkcije. Kroz iteracije algoritma trebamo nastaviti pomicati se u tom smjeru, odnosno ići prema naprijed. Dobijemo li suprotnu nejednakost s obzirom na (8), trebamo promijeniti smjer tj. stavljamo $h_0 = -h_0$, odnosno pomicati se unazad.

Algoritam 1 Algoritam "naprijed-nazad"

Input: $\alpha_0 \in [0, +\infty)$, h_0, $k = 0$, $t > 2$ (najčešće koristimo $t = 2$)
Output: početni interval $I = [a, b]$

Izračunaj $\alpha_1 \leftarrow \alpha_0 + h_0$, $\phi_1 \leftarrow \phi(\alpha_1)$
if $\phi_1 > \phi_0$ **then**
    $h_0 \leftarrow -h_0$
    $\alpha_1 \leftarrow \alpha_0 + h_0$
end if
while $\phi_{k+1} < \phi_k$ **do**
    $h_{k+1} \leftarrow t \cdot h_k$
    $\alpha \leftarrow \alpha_k$, $\alpha_k \leftarrow \alpha_{k+1}$
    $k \leftarrow k + 1$
    $\alpha_{k+1} \leftarrow \alpha_k + h_k$
end while
$a \leftarrow \min\{\alpha, \alpha_{k+1}\}$, $b \leftarrow \max\{\alpha, \alpha_{k+1}\}$
return a,b

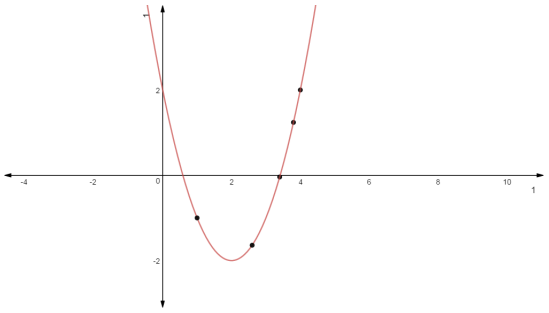
Primjer 3 *Odredimo početni interval funkcije* $\phi(\alpha) = \alpha^2 - 4\alpha + 2$ *koristeći Algoritam "naprijed-nazad", pri čemu je* $\alpha_0 = 4$, $h_0 = 0.1$ *i* $t = 2$.

Ovaj primjer riješit ćemo koristeći Matlab te sljedeći programski kod:

```
function [aa,bb] = pocetniinterval(a0, h0, t, f, maxit)
a1=a0+h0;
    if f(a1)>f(a0)
        h0=-h0;
        a1=a0+h0;
    end
for ii= 1:maxit
```

```
    h1=t*h0;
    a=a0;
    a1=a0+h1;
    h0=h1;
    a0=a1;
    disp(sprintf('%d.iteracija: a0: [%g, %g], a1: [%g, %g] h:%g
       ' , ii, a, f(a), a1, f(a1),h1))
    if f(a1)>=f(a)
        aa=min(a,a1);
        bb=max(a,a1);
        disp(sprintf('I=  [%g, %g]', aa, bb))
        break
    end
end
```

Za $\alpha_0 = 1$, $h_0 = 0.1$ i $t = 2$ rješenje je $[1, 2.6]$. Iteracije algoritma prikazane su u Tablici 1.



Slika 4: grafički prikaz funckije ϕ i iteracija iz Primjera 3

k	α_k	$\phi(\alpha_k)$	h_k
0	4	2	-0.1
1	3.8	1.24	-0.2
2	3.4	-0.04	-0.4
3	2.6	-1.64	-0.8
4	1	-1	-1.6

Tablica 1: Iterativni postupak iz Primjera 3 primjenom Matlab-modula

Kao što možemo vidjeti iz Tablice 1 te sa Slike 4 na početku algoritma mijenjamo smjer tj. stavljamo $h_0 = -h_0 = -0.1$ i pomičemo se unazad kroz četiri iteracije smanjujući vrijednost funkcije ϕ. U četvrtoj iteraciji ($k = 4$) pomicanjem unazad vrijednost funkcije ϕ bi se povećala, stoga algoritam staje i kao rezultat vraća početni interval $[1, 2.6]$.

4 Optimalno linijsko pretraživanje

Kao što je već spomenuto u Poglavlju 1, optimalne metode linijskog pretraživanja temelje se na rješavanju minimizacijskog problema

$$f(x_k + \alpha_k p_k) = \min_{\alpha>0} f(x_k + \alpha p_k). \tag{9}$$

Rješenje prethodnog jednodimenzionalnog optimizacijskog problema daje nam duljinu koraka α_k u datoj iteraciji k. Ako je moguće analitički odrediti minimum $\alpha > 0$ iz (9), onda se u tu svrhu može koristiti neka od metoda optimalnog linijskog pretraživanja, koje ćemo predstaviti u Poglavlju 4. Nažalost, prilikom rješavanja problema u praksi to nije čest slučaj. U tom slučaju ova klasa metoda postaje računski zahtjevna, neefikasna (vremenski zahtjevna, zauzima puno memorije). Metode optimalnog linijskog pretraživanja imaju teorijsku važnost zbog lakšeg razumijevanja te rješavanja daljnjih problema. Također, u okviru Poglavlja 4 predstavit ćemo nekoliko takvih metoda.

4.1 Metode temeljene na podjeli intervala

Metoda zlatnog reza i Fibonaccijeva metoda su metode optimizacije temeljene na podjeli intervala. Želimo minimizirati unimodalnu funkciju $\phi : [a, b] \to \mathbb{R}$ iterativno smanjivajući interval $[a, b]$, na temelju usporedbi vrijednosti koje funkcija postiže u određenim točkama. Iterativni postupak primjenjujemo dok duljina intervala ne postane manja od unaprijed zadane tolerancije. Ovakve metode ne zahtijevaju diferencijabilnost funkcije, stoga su pogodne za nediferencijabilne funkcije, te kod funkcija kojima je njihov izračun računski zahtjevan.

4.1.1 Metoda zlatnog reza

Neka je $\phi(\alpha) = f(x + \alpha p)$ unimodalna funkcija na zatvorenom intervalu $[a, b]$, tražimo $\alpha^* \in [a, b]$. Biramo dvije točke λ_k, $\mu_k \in [a_k, b_k]$ takve da $\lambda_k < \mu_k$ u kojima ćemo izračunati vrijednost funkcije. Kod izbora λ_k i μ_k zahtijevamo da budu zadovoljeni sljedeći uvjeti:

- udaljenost λ_k i μ_k od krajnjih točaka mora biti jednaka

$$\lambda_k - b_k = \mu_k - a_k$$

- λ_k se nalazi u lijevoj polovici intervala, dok se μ_k nalazi u desnoj polovici intervala $[a, b]$, za $c \in (0, \frac{1}{2})$

$$\lambda_k = a_k + c(b_k - a_k)$$

$$\begin{aligned} \mu_k &= b_k - c(b_k - a_k) \\ &= a_k + (1 - c)(b_k - a_k) \end{aligned}$$

Kako je ϕ unimodalna imamo dvije mogućnosti:

1. Ako je $\phi(\lambda_k) \leq \phi(\mu_k)$ onda $a_{k+1} = a_k$, $b_{k+1} = \mu_k$, za $\mu_{k+1} = \lambda_k$ dok nam je aproksimacija rješenja $\alpha^* = \lambda_k$.

2. Ako je $\phi(\lambda_k) > \phi(\mu_k)$ onda $a_{k+1} = \lambda_k$, $b_{k+1} = b_k$, za $\lambda_{k+1} = \mu_k$ dok nam je aproksimacija rješenja $\alpha^* = \mu_k$.

Dobili smo segment $[a_{k+1}, b_{k+1}] \subset [a, b]$ koji je očigledno manji od početnog intervala što je upravo ono što smo željeli i postići. Iterativni postupak provodimo dok udaljenost između rubnih točaka intervala ne bude manja od unaprijed zadane tolerancije. Pretpostavimo da vrijedi $\phi(\lambda_k) \leq \phi(\mu_k)$ u $k+1$ iteraciji imamo

$$\mu_{k+1} = a_{k+1} + (1-c)(b_{k+1} - a_{k+1}) \tag{10}$$
$$\mu_{k+1} = \lambda_k = a_k + c(b_k - a_k). \tag{11}$$

Znamo da vrijedi

$$\mu_k = a_k + (1-c)(b_k - a_k)$$
$$(b_k - a_k) = \frac{1}{1-c}(\mu_k - a_k).$$

Kako je $\phi(\lambda_k) \leq \phi(\mu_k)$, imamo slučaj 1., stoga slijedi $a_{k+1} = a_k$, $b_{k+1} = \mu_k$

$$\mu_{k+1} = a_{k+1} + \frac{c}{1-c}(b_{k+1} - a_{k+1}). \tag{12}$$

Iz (10) i (12) dobivamo $1 - c = \frac{c}{1-c}$, kako je $c \in (0, \frac{1}{2})$ slijedi $c = \frac{3-\sqrt{5}}{2}$. Analogno za slučaj da je $\phi(\lambda_k) > \phi(\mu_k)$.

Algoritam 2 Algoritam zlatnog reza

Input: a_1, b_1, $k = 1$, $\delta > 0$
Output: $\alpha^* \in [a_k, b_k]$

$\lambda_1 \leftarrow a_1 + \frac{3-\sqrt{5}}{2}(b_1 - a_1)$
$\mu_1 \leftarrow a_1 + (1 - \frac{3-\sqrt{5}}{2})(b_1 - a_1)$
while $|b_k - a_k| > \delta$ **do**
    if $\phi(\lambda_k) > \phi(\mu_k)$ **then**
        $a_{k+1} \leftarrow \lambda_k$
        $b_{k+1} \leftarrow b_k$
        $\lambda_{k+1} \leftarrow \mu_k$
        $\alpha^* \leftarrow \mu_k$
        $\mu_{k+1} \leftarrow a_{k+1} + (1 - \frac{3-\sqrt{5}}{2})(b_{k+1} - a_{k+1})$
    else
        $a_{k+1} \leftarrow a_k$
        $b_{k+1} \leftarrow \mu_k$
        $\mu_{k+1} \leftarrow \lambda_k$
        $\alpha^* \leftarrow \lambda_k$
        $\lambda_{k+1} \leftarrow a_{k+1} + \frac{3-\sqrt{5}}{2}(b_{k+1} - a_{k+1})$
    end if
    $k = k + 1$
end while
return α^*

Primjer 4 *Neka je* $\phi : [a, b] \to \mathbb{R}$ *zadana sa* $\phi(\alpha) = \alpha^2 - 4\alpha + 2$. *Odredimo optimalnu duljinu koraka* α^*, *uz tolaranciju* 0.001, *koristeći metodu zlatnog reza. Iz Primjera 3 znamo da je početni interval* $[1, 2.6]$

Za rješavanje primjera koristiti ćemo sljedeći Matlab-modul.

```
function [alfa] = zlatnirez(a1, b1, f, tol, maxit)
lambda1=a1+0.3820*(b1-a1);
mi1=a1+0.6180*(b1-a1);
for ii= 1:maxit
    if f(lambda1)>f(mi1)
        a1=lambda1;
        lambda1=mi1;
        alfa=mi1;
        mi1=a1+0.6180*(b1-a1);
    else
        b1=mi1;
        mi1=lambda1;
        alfa=lambda1;
        lambda1=a1+0.3820*(b1-a1);
    end
    disp(sprintf('%d.iteracija: a1=%g, b1= %g,lambda1=%g, mi1=%
       g, alfa=%g' , ii, a1, b1, lambda1, mi1,alfa))
    if abs(b1-a1)<tol
        disp(sprintf('a= %g', alfa))
        break
    end
end
```

Primjenom navedenog modula dobivamo Tablicu 2. Iz tablice vidimo da nakon 16 iteracija vrijedi $\alpha^* = 1.99993$, dok je pogreška $|\alpha^* - 2| = 0.00007$.

k	a_k	b_k	λ_k	μ_k	α^*
0	1	2.6	1.6112	1.9888	1.9888
1	1.6112	2.6	1.9888	2.22228	1.9888
2	1.6112	2.22228	1.84463	1.9888	1.9888
3	1.84463	2.22228	1.9888	2.07802	1.9888
4	1.84463	2.07802	1.93379,	1.9888	1.9888
5	1.93379	2.07802	1.9888	2.02292	1.9888
6	1.93379	2.02292	1.96783	1.9888	1.9888
7	1.96783	2.02292	1.9888	2.00188	1.9888
8	1.9888	2.02292	2.00188	2.00989	2.00188
9	1.9888	2.00989	1.99686	2.00188	2.00188
10	1.99686	2.00989	2.00188	2.00491	2.00188
11	1.99686	2.00491	1.99993	2.00188	2.00188
12	1.99686	2.00188	1.99877	1.99993	1.99993
13	1.99877	2.00188	1.99993	2.00069	1.99993
14	1.99877	2.00069	1.99951	1.99993	1.99993
15	1.99951	2.00069	1.99993	2.00024	1.99993
16	1.99951	2.00024	1.99979	1.99993	1.99993

Tablica 2: Iterativni postupak iz Primjera 4 primjenom Matlab-modula za zlatni rez

4.1.2 Fibonaccijeva metoda

Fibonaccijeva metoda vrlo je slična metodi zlatnog reza, štoviše metoda zlatnog reza je specijalan slučaj Fibonaccijeve metode. Razlika između tih dviju metoda je ta da kod Fibonaccijeve metode za redukciju intervala koristimo Fibonaccijev niz, stoga nemamo uvijek isti koeficijent redukcije intervala u svakoj iteraciji. Prilikom izvođenja Fibonaccijeve metode moramo unaprijed znati broj koraka. Definirajmo Fibonaccijev niz.

Definicija 2 *Neka je* $F_0 = 1$, $F_1 = 1$. *Niz* (F_n) *definiran rekurzivnom relacijom*

$$F_n = F_{n-1} + F_{n-2},$$

za $n \geq 2$ *naziva se Fibonaccijev niz. Član niza* F_n *zove se* n*-ti Fibonaccijev broj.*

Kod metode zlatnog reza za koeficijent redukcije intervala koristi se $\frac{3-\sqrt{5}}{2}$, dok ćemo kod Fibonaccijeve metode koristiti $\frac{F_{n-k}}{F_{n-k+1}}$. Uočimo da je

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

Zato slijedi

$$\lambda_k = a_k + \frac{F_{n-k-1}}{F_{n-k+1}}(b_k - a_k), k = 1, \ldots, n-1$$
$$\mu_k = a_k + \frac{F_{n-k}}{F_{n-k+1}}(b_k - a_k), k = 1, \ldots, n-1.$$

Kada $k \to \infty$ metoda zlatnog reza i Fibonaccijeva metoda imaju isti koeficijent redukcije intervala.

Primjer 5 *Neka je* $\phi : [a, b] \to \mathbb{R}$ *zadana sa* $\phi(\alpha) = \alpha^2 - 4\alpha + 2$. *Odredimo optimalnu duljinu koraka* α^*, *koristeći Fibbonacijevu metodu ako je zadan broj iteracija* $k = 16$. *Iz Primjera 3 znamo da je početni interval* $[1, 2.6]$.

Za rješavanje primjera koristit ćemo sljedeći Matlab-modul.

```
function [alfa] = fibonaccijevametoda(a1, b1, f, n)
fib(1)=1;
fib(2)=1;
for jj=3:n
    fib(jj)=fib(jj-1)+fib(jj-2);

end
lambda1=a1+fib(n-2)/fib(n)*(b1-a1);
mi1=a1+fib(n-1)/fib(n)*(b1-a1);
for ii= 1:n-2
    if f(lambda1)>f(mi1)
        a1=lambda1;
        lambda1=mi1;
        alfa=mi1;
        mi1=a1+fib(n-ii)/fib(n)*(b1-a1);
```

```
    else
        b1=mi1;
        mi1=lambda1;
        alfa=lambda1;
        lambda1=a1+fib(n-ii-1)/fib(n)*(b1-a1);
    end
    disp(sprintf('%d.iteracija: a1=%g, b1= %g,lambda1=%g, mi1=%
       g, alfa=%g' , ii, a1, b1, lambda1, mi1,alfa))

end
        disp(sprintf('a= %g', alfa))
end
```

Nakon 16 iteracija ($n = 18$) dobivamo da je $\alpha^* = 1.98885$. Uočimo, s obzirom na metodu zlatnog reza, da smo dobili manje precizan rezultat, tj. veća je pogreška $|\alpha^* - 2| = 0.0111$.

4.2 Metode interpolacije

Ova klasa metoda optimizacije aproksimira zadanu funkciju, u našem slučaju $\phi(\alpha) = f(x + \alpha p)$, kvadratnim ili kubnim polinomom s kojim je lakše računati. Za zadane točke $x_1, \ldots, x_k$ tražimo polinom interpolacije $P_n = a_n x^n + \cdots + a_1 x_1 + a_0$ tako da vrijedi $P_n(x_i) = \phi(x_i), i = 1, \ldots, k$. Funkcija ϕ i interpolacijski polinom P_n podudaraju se u zadanim točkama $x_1, \ldots, x_k$.

4.2.1 Kvadratna interpolacija

Kvadratna interpolacija sa dvije zadane točke

Neka su zadane točke α_1, α_2, njihove funkcijske vrijednosti $\phi(\alpha_1)$, $\phi(\alpha_2)$ i derivacija $\phi'(\alpha_1)$ (analogno bi postupali da je zadano $\phi'(\alpha_2)$). Označimo $\phi(\alpha_1) := \phi_1$, $\phi(\alpha_2) := \phi_2$, te $\phi'(\alpha_1) = \phi_1'$ radi lakšeg zapisa. Problem interpolacije svodi se na pronalazak kvadratnog interpolacijskog polinoma $P_2(\alpha) = A\alpha^2 + B\alpha + C$, odnosno koeficijenata A, B, C. Znamo da vrijedi:

$$P_2(\alpha_1) = A\alpha_1^2 + B\alpha_1 + C = \phi_1 \tag{13}$$
$$P_2(\alpha_2) = A\alpha_2^2 + B\alpha_2 + C = \phi_2 \tag{14}$$
$$P_2'(\alpha_1) = 2A\alpha_1 + B = \phi_1'. \tag{15}$$

Iz (15) slijedi da je $B = \phi_1' - 2A\alpha_1$. Nadalje, od jednadžbe (13) oduzmimo jednadžbu (14), te uvrstimo B. Slijedi

$$A = \frac{\phi_1 - \phi_2 - \phi_1'(\alpha_1 - \alpha_2)}{-(\alpha_1 - \alpha_2)^2}$$

odnosno

$$B = \phi_1' - 2\frac{\phi_1 - \phi_2 - \phi_1'(\alpha_1 - \alpha_2)}{-(\alpha_1 - \alpha_2)^2}\alpha_1.$$

Dakle,

$$\begin{aligned}\alpha^* &= -\frac{B}{2A} \\ &= \alpha_1 - \frac{1}{2}\frac{(\alpha_1 - \alpha_2)\phi_1'}{\phi_1' - \frac{\phi_1 - \phi_2}{\alpha_1 - \alpha_2}}.\end{aligned}$$

Razmišljamo li u terminima iteracije dobivamo formulu

$$\alpha^*_{k+1} = \alpha_{k-1} - \frac{1}{2}\frac{(\alpha_{k-1} - \alpha_k)\phi'_{k-1}}{\phi'_{k-1} - \frac{\phi_{k-1}-\phi_k}{\alpha_{k-1}-\alpha_k}}.$$

Ideja metode ja da nakon izračuna α_{k+1} usporedimo ga s već poznatim α_k i α_{k-1}, te smanjimo početni interval. Postupak ponavljamo dok nam interval ne postane manji od neke unaprijed zadane tolerancije.

Kvadratna interpolacija sa tri zadane točke

Neka su zadane redom točke α_1, α_2, α_3 i njihove funkcijske vrijednosti $\phi(\alpha_1)$, $\phi(\alpha_2)$, $\phi(\alpha_3)$. Želimo konstruirati kvadratni interpolacijski polinom $P_2(\alpha) = A\alpha^2 + B\alpha + C$ tako da vrijedi

$$P_2(\alpha_i) = A\alpha_i^2 + B\alpha_i + C = \phi(\alpha_i), i = 1, 2, 3.$$

Zbog lakšeg zapisa označimo $\phi(\alpha_i) := \phi_i$, $i = 1, 2, 3$. Definirajmo funkcije

$$L_1(\alpha) = \frac{(\alpha - \alpha_2)(\alpha - \alpha_3)}{(\alpha_1 - \alpha_2)(\alpha_1 - \alpha_3)} \tag{16}$$

$$L_2(\alpha) = \frac{(\alpha - \alpha_1)(\alpha - \alpha_3)}{(\alpha_2 - \alpha_1)(\alpha_1 - \alpha_3)} \tag{17}$$

$$L_3(\alpha) = \frac{(\alpha - \alpha_1)(\alpha - \alpha_2)}{(\alpha_3 - \alpha_1)(\alpha_3 - \alpha_2)}. \tag{18}$$

Definirajmo Lagrangeov interpolacijski polinom.

Definicija 3 *Kvadratni Lagrangeov interpolacijski polinom kroz točke* (α_1, ϕ_1), (α_2, ϕ_2) *i* (α_3, ϕ_3) *tako da* $\alpha_1 \neq \alpha_2 \neq \alpha_3$ *je*

$$P_2(\alpha) = \phi_1 L_1(\alpha) + \phi_2 L_2(\alpha) + \phi_3 L_3(\alpha).$$

Stavljamo $P'_2(\alpha) = 0$ te iz jednadžbe izračunavamo α. Kroz sljedeći algoritam prikažimo sve prethodne korake.

Algoritam 3 Kvadratna interpolacija sa tri točke

Input: $\alpha_1 < \alpha_2 < \alpha_3$, $\delta > 0$, k dovoljno velik
Output: α^*

for $i = 1$, i++, $i < k$ **do**
  Izračunati α^* iz $P'_2(\alpha^*) = 0$
  if $(\alpha^* - \alpha_1)(\alpha^* - \alpha_3) \geq 0$ **then**
    Od točaka α_1, α_2, α_2, α^* izaberemo onu s najmanjom funkcijskom vrijednosti te ju označimo s α_2, a za nove α_1 i α_3 stavljamo njezine neposredne susjede
  else
    if $|\alpha^* - \alpha_2| < \delta$ **then**
      return α^*
    else
      Od točaka α_1, α_2, α_2, α^* izaberemo onu s najmanjom funkcijskom vrijednosti te ju označimo s α_2, a za nove α_1 i α_3 stavljamo njezine neposredne susjede
    end if
  end if
end for

Primjer 6 *Neka je* $\phi : [0.5, 5] \to \mathbb{R}$ *zadana sa* $\phi(\alpha) = \alpha + \frac{6}{\alpha+1}$. *Odredimo optimalnu duljinu koraka* α^*, *koristeći metodu kvadratne interpolacije. Neka je tolerancija* 0.01, *a za zadane dvije točke uzimamo rubne točke intervala na kojem je funkcija definirana.*

Za rješavanje koristit ćemo sljedeći Matlab-modul.

```
function [aproks] = kvadratnainterpolacija(a1, a2, f,df, tol,
   maxit)
x0=(a1+a2)/2;
for ii= 1:maxit
    x1=a1-0.5*((a1-a2)*df(a1))/(df(a1)-(f(a1)-f(a2))/(a1-a2));
    mn=min(x0,x1);
    mx=max(x0,x1);
    if abs(x0-x1)<tol
        disp(sprintf('k= %g, alfa1=%g, alfa2=%g, alfa*= %g', ii
           , a1, a2 ,x1))
        break
    end
    if f(mx)<f(mn)
        a1=mn;
        x0=mx;
    else
        a2=mx;
        x0=mn;

    end
disp(sprintf('k= %g, alfa1=%g, alfa2=%g, alfa*= %g', ii, a1, a2
    ,x1))

end
```

Primjenom navedenog modula nakon šest iteracija dobivamo $\alpha^* = 1.45162$ uz zadanu toleranciju 0.01.

k	α_1	α_2	α^*
1	0.5	2.75	2.375
2	0.5	2.375	1.67188
3	0.5	1.67188	1.55469
4	0.5	1.67188	1.33496
5	0.5	1.55469	1.45697
6	0.5	1.55469	1.45162

Tablica 3: Iterativni postupak iz Primjera 8 primjenom Matlab-modula za kubnu interpolaciju

4.2.2 Kubna interpolacija

Početna ideja kod kvadratne i kubne interpolacije ne razlikuje se. Promjena s obzirnom na kvadratnu interpolaciju je da imamo polinom trećeg stupnja. Kod kubne interpolacije moramo imati zadovoljena četiri uvjeta (dok kod kvadratne moramo imati tri) npr. vrijednosti

funkcije kod 4 zadane točke, vrijednosti funkcije kod 3 zadane točke, te vrijednost derivacije kod jedne od njih, itd. Općenito, može se dokazati da kubna interpolacija brže konvergira nego kvadratna, ali računski je više zahtjevna.

Kubna interpolacija sa zadane 2 točke

Neka su zadane dvije točke α_1, α_2, njihove funkcijske vrijednosti $\phi(\alpha_1)$, $\phi(\alpha_2)$, te derivacije $\phi'(\alpha_1)$, $\phi'(\alpha_2)$. Označimo $\phi(\alpha_1) := \phi_1$, $\phi(\alpha_2) := \phi_2$, te $\phi'(\alpha_1) := \phi'_1$, $\phi'(\alpha_2) := \phi'_2$ radi lakšeg zapisa. Želimo konstruirati kubni interpolacijski polinom $P_3(\alpha) = A(\alpha-\alpha_1)^3+B(\alpha-\alpha_1)^2+C(\alpha-\alpha_1)+D$. Derivacijom P_3 dobivamo $P'_3(\alpha) = 3A(\alpha-\alpha_1)^2+2B(\alpha-\alpha_1)+C$, te slijedi da je

$$\begin{aligned} P_3(\alpha_1) &= D = \phi_1 \\ P'_3(\alpha_1) &= C = \phi'_1 \\ P_3(\alpha_2) &= A(\alpha_2-\alpha_1)^3 + B(\alpha_2-\alpha_1)^2 + C(\alpha_2-\alpha_1) + D = \phi_2 \\ P'_3(\alpha_2) &= 3A(\alpha_2-\alpha_1)^2 + 2B(\alpha_2-\alpha_1) + C = \phi'_2. \end{aligned} \tag{19}$$

Rješavanjem kvadratne jednadžbe $P'_3(\alpha) = 0$ dobivamo:

$$\alpha - \alpha_1 = \frac{-B \pm \sqrt{B^2-3AC}}{3A}, \qquad za\ A \neq 0 \tag{20}$$

$$\alpha - \alpha_1 = -\frac{C}{2B}, \qquad za\ A = 0. \tag{21}$$

Da bi α bio minimum funkcije P_3 mora vrijediti da je $P'_3(\alpha) = 0$ što smo računali u (20) i (21), te mora vrijediti da je $P''_3(\alpha) > 0$. Stoga promotrimo P''_3

$$P''_3(\alpha) = 6A(\alpha-\alpha_1) + 2B > 0.$$

Vidimo da $(\alpha - \alpha_1) > \frac{-B}{3A}$, zato kod (20) uzimamo u obzir samo pozitivnu vrijednost. Kombinacijom (20) i (22), odnosno korištenjem činjenice da je $B = \sqrt{B^2-3AC}$ slijedi nam

$$\alpha^* = \alpha_1 - \frac{C}{B+\sqrt{B^2-3AC}}. \tag{22}$$

Za $A = 0$ koristimo samo (21). Reprezentirajmo α^* pomoću poznatih vrijednosti $\phi(\alpha_1)$, $\phi(\alpha_2)$, $\phi'(\alpha_1)$, $\phi'(\alpha_2)$. Neka su

$$\begin{aligned} s &= 3\frac{\phi_2-\phi_1}{\alpha_2-\alpha_1} \\ z &= s - \phi'_1 - \phi'_2 \\ w^2 &= z^2 - \phi'_1\phi'_2. \end{aligned}$$

Iskoristimo li (19) imamo

$$\begin{aligned} s &= 3\frac{\phi_2-\phi_1}{\alpha_2-\alpha_1} = 3A(\alpha_2-\alpha_1)^2 + 3B(\alpha_2-\alpha_1) + 3C \\ z &= s - \phi'_1 - \phi'_2 = B(\alpha_2-\alpha_1) + C \\ w^2 &= z^2 - \phi'_1\phi'_2 = (\alpha_2-\alpha_1)^2(B^2-3AC). \end{aligned}$$

Zatim iz druge i treće jednadžbe slijedi

$$(\alpha_2 - \alpha_1)B = z - C \qquad i \qquad \sqrt{B^2 - 3AC} = \frac{w}{\alpha_2 - \alpha_1}$$

Odnosno imamo $B + \sqrt{B^2 - 3AC} = \frac{z+w-C}{\alpha_2-\alpha_1}$ i kako znamo da je $C = \phi_1'$ iz (22) slijedi da je

$$\alpha^* = \alpha_1 + \frac{(\alpha_2 - \alpha_1)(w - z)}{\phi_2' - z + w}. \tag{23}$$

Primjer 7 *Neka je* $\phi : [0.5, 5] \to \mathbb{R}$ *zadana sa* $\phi(\alpha) = \alpha + \frac{6}{\alpha+1}$. *Odredimo optimalnu duljinu koraka* α^* *koristeći metodu kubne interpolacije. Neka je tolerancija* 0.01, *a za zadane dvije točke uzimamo rubne točke intervala na kojem je funkcija definirana.*

Za rješavanje koristit ćemo sljedeći Matlab-modul.

```
function [alfa] = kubnainterpolacija(a, b, f, df, tol, maxit)
z=3*(f(b)-f(a))/(b-a);
w=sqrt(z^2-df(a)*df(b));
a1=a+((b-a)*(w-z))/(df(b)-z+w);
for ii= 1:maxit
    if df(a1)<0
        a=a1;
    else
        b=a1;
    end
    z=3*(f(b)-f(a))/(b-a);
    w=sqrt(z^2-df(a)*df(b));
    aproks=a+((b-a)*(w-z))/(df(b)-z+w);
    if abs(aproks-a1)<tol
                disp(sprintf('a= %g', aproks))
        break;
    end
    disp(sprintf('k= %g alfa= %g a= %g b= %g',ii, aproks,a,b))
    a1=aproks;
end
```

Nakon devet iteracija dobivamo $\alpha^* = 1.48489$ uz zadanu toleranciju 0.01. Općenito, kubna interpolacija brže konvergira nego kvadratna. Kao što možemo vidjeti i na ovome primjeru ako se radi o malom broju iteracija ponekad je efikasnije koristiti kvadratnu interpolaciju.

k	a_k	b_k	α_k
1	0.5	2.28051	1.9195
2	0.5	1.9195	1.765
3	0.5	1.765	1.67495
4	0.5	1.67495	1.61606
5	0.5	1.61606	1.57512
6	0.5	1.57512	1.54556
7	0.5	1.54556	1.52367
8	0.5	1.52367	1.50718
9	0.5	1.50718	1.48489

Tablica 4: Iterativni postupak iz Primjera 7 primjenom Matlab-modula za kubnu interpolaciju

5 Aproksimativno linijsko pretraživanje

Iterativni oblik rješenja kod ovog tipa linijskog pretraživanja ne razlikuje se od optimalnog linijskog pretraživanja i glasi

$$x_{k+1} = x_k + \alpha_k \, p_k.$$

Pritom je $\alpha_k > 0$ duljina koraka, a $p_k \in \mathbb{R}^n$ vektor smjera. Umjesto egzaktne minimizacije po duljini koraka, u ovom pristupu zahtijevamo samo smanjenje vrijednosti funkcije unutar svakog iterativnog koraka, tj. da vrijedi

$$f(x_k + \alpha_k p_k) < f(x_k). \tag{24}$$

U ovom poglavlju opisat ćemo metode aproksimativnog linijskog pretraživanja kojima je cilj postići što optimalnije rješenje. Prvo postavimo nekoliko uvjeta na funkciju $f : \mathbb{R}^n \to \mathbb{R}$. Neka je f neprekidno diferencijabilna, želimo aproksimativno riješiti problem

$$\alpha^* = \underset{\alpha>0}{\operatorname{argmin}} \, f(x_k + \alpha p_k)$$

pri čemu je x_k trenutna iteracija, a p_k vektor smjera takav da zadovoljava $\nabla f(x_k)^T p_k < 0$. Ako vrijedi $\nabla f(x_k)^T p_k < 0$ onda iz Teorema 1 slijedi da je $p_k \in \mathbb{R}^n$ silazni vektor smjera, tj. vrijedi (24). U terminima funkcije ϕ to izgleda ovako

$$\phi(\alpha) = f(x_k + \alpha p_k).$$

Slično kao i u Poglavlju 4 želimo izračunati α^* iz $\phi'(\alpha^*)$. Znamo da je

$$\phi'(\alpha) = \nabla f(x_k + \alpha p_k)^T p_k.$$

Nadalje, kako vrijedi Teorem 1 zaključujemo da je $\phi'(\alpha) < 0$.

5.1 Armijevo pravilo i linijsko pretraživanje unazad

Unutar aproksimativnog linijskog pretraživanja želimo da nam izbor duljine koraka α_k daje dovoljno smanjenje funkcije f. Prvo ćemo promotriti tzv. Armijevo pravilo. Neka su $0 < \mu \leq \frac{1}{2}$ i $\beta \in (0,1)$, $\tau > 0$, gdje je m_k najmanji cijeli broj za kojeg vrijedi nejednakost

$$f(x_k) - f(x_k + \beta^m \tau p_k) \geq -\mu \beta^m \tau \nabla f(x_k)^T p_k. \tag{25}$$

Za nađenu vrijednost m_k uz oznaku $\alpha := \mu \beta^m \tau$, ovo pravilo preformuliramo te imamo

$$f(x_k) - f(x_k + \alpha p_k) \geq -\mu \alpha \nabla f(x_k)^T p_k.$$

Odnosno

$$f(x_k + \alpha p_k) - f(x_k) \leq \mu \alpha \nabla f(x_k)^T p_k. \tag{26}$$

Nejednadnakost (26) daje nam takozvanu metodu linijskog pretraživanja unatrag. Ideja ove metode je da uzmemo prihvatljivi $\alpha > 0$, tj. α koji je dovoljno daleko do 0, a koji zadovoljava (26). Početno postavimo $\alpha = 1$, te ako $x_k + \alpha p_k$ nije prihvatljiv, reduciramo α dok izraz ne zadovoljava (26).

Algoritam 4 Algoritam linijskog pretraživanja unatrag

Input: $\mu \in (0, \frac{1}{2}]$, $\beta \in (0, 1)$
Output: α_k, x_{k+1}
1: $\alpha \leftarrow 1$
2: **while** $f(x_k + \alpha p_k) - f(x_k) > \mu\alpha\nabla f(x_k)^T p_k$ **do**
3: $\quad \alpha \leftarrow \beta\alpha$
4: **end while**
5: **return** $\alpha_k = \alpha$, $x_{k+1} = x_k + \alpha_k p_k$

Iz algoritma vidimo čemu naziv pretraživanje unatrag. Krećemo s prvom iteracijom gdje je vrijednost $\alpha = 1$ te ju iterativno smanjujemo (u smjeru unatrag) dok uvjet (26) ne bude zadovoljen. Vidimo da je algoritam poprilično jednostavan, kao i njegova implementacija stoga je često korišten u praksi.

Primjer 8 *Neka je* $f(x, y) = e^x + x(1+x) + 2y - xy + y^2 + 1$, $x_0 = (4, 2)$ *početna aproksimacija. Gradijentnom metodom uz toleranciju* 0.01*, koristeći linijsko pretraživanje unatrag riješimo optimizacijski problem ako je* $\beta = \frac{1}{4}$, *a* $\mu = \frac{1}{2}$.

Za rješavanje koristit ćemo sljedeći Matlab-modul.

```
function [x] = gmlinunatrag(f, df, x0, tol, maxit, c)

for ii=1:maxit
    pk = -df(x0);
    %Algoritam linijskog pretrazivanja unatrag
    alfa=1;
    for jj=1:maxit
        if f(x0+alfa*pk)-f(x0)<=0.5*alfa*df(x0)'*pk
            break
        end
        alfa=c*alfa;
    end
    x = x0+alfa*pk;
    x0 = x;
    disp(sprintf('%d.iteracija: x* = [%f %f], greska: %g, alfak
       = %g', ii, x0, norm(df(x0)), alfa))
    if norm(df(x0))<tol
        break
    end
end
```

Nakon 21. iteracije dobivamo aproksimaciju rješenja $x^* = (-1.4796, -1.7368)$ uz toleranciju 0.01. Unutar Tablice 6 zapisane su vrijednosti α_k unutar svake iteracije.

k	(x_k, y_k)	$\|\nabla f(x_k, y_k)\|$	α_k
1	(3.037529, 1.968750)	26.1214	$\frac{1}{64}$
2	(1.415032 ,1.787502)	7.43242	$\frac{1}{16}$
3	(-0.124763 ,0.747509)	3.72656	$\frac{1}{4}$
4	(-0.346181 ,-0.157436)	2.34539	$\frac{1}{4}$
5	(-0.639296 ,-0.665263)	1.59652	$\frac{1}{4}$
6	(-0.867880 ,-0.992456)	1.11236	$\frac{1}{4}$
7	(-1.037014 ,-1.213198)	0.785224	$\frac{1}{4}$
8	(-1.160434 ,-1.365852)	0.55876	$\frac{1}{4}$
9	(-1.250018 ,-1.473035)	0.399655	$\frac{1}{4}$
10	(-1.314892 ,-1.549022)	0.286827	$\frac{1}{4}$
11	(-1.361827 ,-1.603234)	0.206326	$\frac{1}{4}$
12	(-1.395770 ,-1.642074)	0.148654	$\frac{1}{4}$
13	(-1.420314 ,-1.669980)	0.107221	$\frac{1}{4}$
14	(-1.438061 ,-1.690068)	0.0773974	$\frac{1}{4}$
15	(-1.450895 ,-1.704550)	0.0559001	$\frac{1}{4}$
16	(-1.460175 ,-1.714998)	0.0403898	$\frac{1}{4}$
17	(-1.466886 ,-1.722543)	0.0291913	$\frac{1}{4}$
18	(-1.471739 ,-1.727993)	0.0211021	$\frac{1}{4}$
19	(-1.475249 ,-1.731931)	0.0152567	$\frac{1}{4}$
20	(-1.477788 ,-1.734778)	0.0110317	$\frac{1}{4}$
21	(-1.479624 ,-1.736836)	0.00797729	$\frac{1}{4}$

Tablica 5: Iterativni postupak iz Primjera 8

5.2 Goldsteinovo pravilo

Također želimo da se $\alpha \in [a, b]$ dovoljno smanjuje, ali uz oprez da je α dovoljno daleko do 0. Kako bi garantirali smanjenje funkcije f, želimo da nam α bude udaljen od krajnjih točaka početnog intervala a i b. Koristimo Armijevo pravilo te još jednu dodatnu nejednakost, gdje je $\mu \in (0, \frac{1}{2}]$

$$f(x_k + \alpha p_k) \leq f(x_k) + \mu\alpha\nabla f(x_k)^T p_k \tag{27}$$

$$f(x_k + \alpha p_k) \geq f(x_k) + (1-\mu)\alpha\nabla f(x_k)^T p_k. \tag{28}$$

Pogledajmo kako to izgleda u terminima funkcije ϕ

$$\phi(\alpha_k) \leq \phi(0) + \mu\alpha_k\phi'(0) \tag{29}$$

$$\phi(\alpha_k) \geq \phi(0) + (1-\mu)\alpha_k\phi'(0). \tag{30}$$

Uočimo da je Goldsteinovo pravilo poopćenje Arimijevog pravila sa "jačim" uvjetima.

Algoritam 5 Algoritam Goldsteinove metode

Input: $\mu \in (0, \frac{1}{2}]$, $\alpha_0 \in [0, \alpha_{max})$, $\epsilon \in [0, 1)$, m dovoljno velik
Output: α_k

$a_0 \leftarrow 0$
$b_0 \leftarrow \alpha_{max}$
$k \leftarrow 0$
for $i = 1$, $i{+}{+}$, $i < m$ **do**
  if $\phi(\alpha_k) \leq \phi(0) + \mu\alpha_k\phi'(0)$ **then**
    if $\phi(\alpha_k) \geq \phi(0) + (1-\mu)\alpha_k\phi'(0)$ **then**
      return α_k
    else
      $a_{k+1} \leftarrow \alpha_k$
      $b_{k+1} \leftarrow b_k$
    end if
  else
    $a_{k+1} \leftarrow a_k$
    $b_{k+1} \leftarrow \alpha_k$
  end if
  $\alpha_{k+1} \leftarrow \frac{a_{k+1}+b_{k+1}}{2}$
  $k = k + 1$
end for

Primjer 9 *Neka je* $f(x, y) = e^x + x(1+x) + 2y - xy + y^2 + 1$, *te neka je* $x_0 = (4, 2)$ *početna aproksimacija,* $\beta = \frac{1}{4}$, *a* $\mu = \frac{1}{2}$. *Prilikom rješavanja optimizacijskog problema koristit ćemo gradijentnu metodu uz toleranciju* 0.01, *dok za izbor duljine koraka koristimo Goldsteinovu metodu.*

Za rješavanje koristit ćemo sljedeći Matlab-modul.

```
function [x] = gmgoldstein(f, df, x0, tol, maxit, amax, m)

for ii=1:maxit
   pk = -df(x0);
   %Goldstein metoda
   a0=0;
   b0=amax;
   for jj=1:maxit
       alfa=a0+b0/2;
       if f(x0+alfa*pk)-f(x0)<=m*alfa*df(x0)'*pk
           if f(x0+alfa*pk)-f(x0)>=(1-m)*alfa*df(x0)'*pk
               break;
           else
               a0=alfa;
           end
       else
           b=alfa;
       end
   end
   x = x0+alfa*pk;
   x0 = x;
```

```
22      disp(sprintf('%d.iteracija: x* = [%f ,%f], greska: %g, alfak
            = %g', ii, x0, norm(df(x0)), alfa))
23      if norm(df(x0))<tol
24          break
25      end
26  end
```

Za razliku od Algoritma 4 kod Goldsteinove metode potrebno je manje iteracija, točnije 16. Aproksimacija rješenja $x^* = (-1.482928, -1.743792)$ nešto je točnija jer imamo manju pogrešku $\|\nabla f(x^*)\| = 0.00676421$. Na Slici 5 možemo vidjeti ponašanje $\|\nabla f(x_k, y_k)\|$ u odnosu na broj iteracija kod obje metode. Kod Algoritma unatrag potreban je veći broj iteracija, točnije $k = 21$, u usporedbi s Algoritmom Goldsteinove metode. Također uočimo da kod Algoritma unatrag početna pogreška unutar prve iteracije je znatno manja. Možemo zaključiti da Algoritmom Goldsteinove metode u ovom slučaju brže konvergira.

k	(x_k, y_k)	$\|\nabla f(x_k, y_k)\|$	α_k
1	(-26.799075 ,1.000000)	61.817	$\frac{1}{2}$
2	(-0.000000 ,-14.39953)	31.4187	$\frac{1}{2}$
3	(-8.199769 ,-1.000000)	16.5703	$\frac{1}{2}$
4	(-1.000137 ,-5.099884)	8.47306	$\frac{1}{2}$
5	(-3.233857 ,-1.500069)	4.51891	$\frac{1}{2}$
6	(-1.269737 ,-2.616928)	2.38808	$\frac{1}{2}$
7	(-1.948917 ,-1.634868)	1.3103	$\frac{1}{2}$
8	(-1.388648 ,-1.974458)	0.71647	$\frac{1}{2}$
9	(-1.611935 ,-1.694324)	0.398481	$\frac{1}{2}$
10	(-1.446913 ,-1.805968)	0.221293	$\frac{1}{2}$
11	(-1.520632 ,-1.723456)	0.123619	$\frac{1}{2}$
12	(-1.471015 ,-1.760316)	0.0690195	$\frac{1}{2}$
13	(-1.495004 ,-1.735508)	0.0386096	$\frac{1}{2}$
14	(-1.479878 ,-1.747502)	0.0215951,	$\frac{1}{2}$
15	(-1.487584 ,-1.739939)	0.0120863	$\frac{1}{2}$
16	(-1.482928 ,-1.743792)	0.00676421	$\frac{1}{2}$

Tablica 6: Iterativni postupak iz Primjera 9

5.3 Wolfe-Powell pravilo

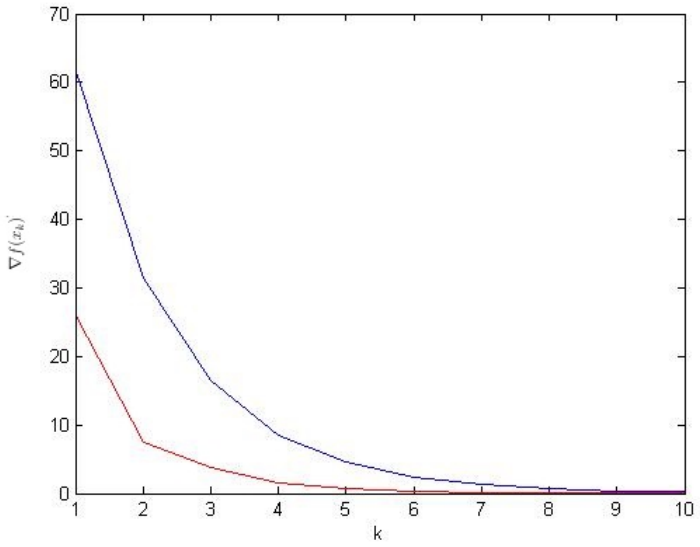
Kod Goldsteinovog pravila postoji mogućnost da uvjet (31) isključi minimalnu vrijednost α kao što je prikazano na Slici 6. Taj problem nam rješava upravo Wolfe-Powell uvjet:

$$(\nabla f(x_k + \alpha p_k))^T p_k \geq \sigma \nabla f(x_k)^T p_k, \qquad \sigma \in (\mu, 1). \tag{31}$$

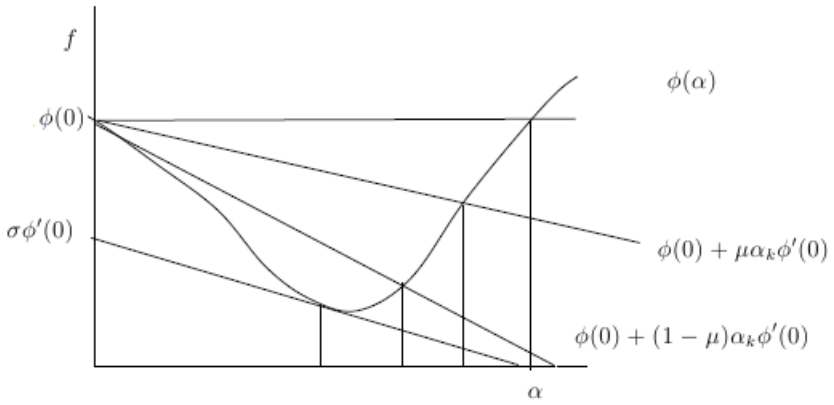
Odnosno,

$$\phi'(\alpha_k) = [\nabla f(x_k + \alpha p_k)]^T p_k \geq \sigma \nabla f(x_k)^T p_k = \sigma \phi'(0) > \phi'(0).$$

Uvjeti (26) i (31) daju nam Wolfe-Powell aproksimativno linijsko pretraživanje ili kraće Wolfe-Powell pravilo. Ova metoda kao rezultat ima interval I koji sadržava minimalnu



Slika 5: Grafički prikaz $\|\nabla f(x_k, y_k)\|$ kroz iteracije, pri čemu je crvena linija dobivena korištenjem Algoritma unatrag, a plava Algoritma Goldsteinove metode



Slika 6: Aproksimativne metode linijskog pretraživanja

vrijednost. Uvjet (31) možemo izvesti iz Goldsteinova pravila te teorema srednje vrijednosti. Neka je α_k takav da vrijedi (35), prema teoremu srednje vrijednosti, pri čemu je $\theta \in (0,1)$, slijedi

$$\alpha_k[\nabla f(x_k + \theta_k \alpha_k p_k)]^T p_k = f(x_k + \alpha_k p_k) - f(x_k) \geq (1-\mu)\alpha_k \nabla f(x_k) T p_k. \tag{32}$$

Neka je $\hat{\alpha}_k$ takav da vrijedi (26), prema teoremu srednje vrijednosti slijedi

$$\hat{\alpha}_k[\nabla f(x_k + \theta_k \hat{\alpha}_k p_k)]^T p_k = f(x_k + \hat{\alpha}_k p_k) - f(x_k) = \mu \hat{\alpha}_k \nabla . f(x_k) T p_k \tag{33}$$

Znamo da $\nabla f(x_k) T p_k < 0$ kako bi osigurali da vrijedi $f(x + \alpha p) < f(x)$. Neka je $\mu < \sigma < 1$ slijedi

$$[\nabla f(x_k + \theta_k \hat{\alpha}_k p_k)]^T p_k = \mu \nabla f(x_k) T p_k > \sigma \nabla f(x_k) T p_k. \tag{34}$$

Uočimo, napravimo li supstituciju $\alpha_k = \theta_k \hat{\alpha}_k$ dobivamo nejednakost (31). Općenito, manja vrijednost parametra σ daje precizniji rezultat. Uglavnom se koristi $\sigma = 0.1$ jer sama vrijednost ne smije biti premala kako si ne bi otežali računanje (povećan broj koraka, vremenski puno duže).

Primjedba: *Primjenimo li na funkciju $f(x,y) = e^x + x(1+x) + 2y - xy + y^2 + 1$ iz Primjera 9 Wolfe-Powell metodu dobit ćemo isti rezultat (isti broj iteracija) kao i kod Goldstein metode.*

5.4 Snažno Wolfe-Powell pravilo

Snažno Wolfe-Powell pravilo zahtjeva dva uvjeta: uvjet (26), te

$$|[\nabla f(x_k + \alpha p_k)]^T p_k| \leq -\sigma \nabla f(x_k)^T p_k, \qquad \sigma \in (\mu, 1). \tag{35}$$

U usporedbi sa Wolfe-Powell pravilom uvjet (35) ne zahtjeva samo da se graf $\phi(\alpha)$ za $\alpha > 0$ ne smanjuje tako strmo kao u $\alpha = 0$, već i da se ne povećava strmo. Kod Wolfe-Powell pravila može se pokazati da kada $\sigma \to 0$ tada postoji mogućnost da duljina koraka α zadovolji uvjet, ali da zapravo nije blizu minimuma funkcije ϕ. Korištenjem Snažnog Wolfe-Powell pravila rješavamo taj problem te isključujemo sve točke koje su daleko od stacionarne točke funkcije ϕ.

5.5 Konvergencija metoda aproksimativnog linijskog pretraživanja

Navedimo općeniti algoritam minimizacije sa aproksimativnom linijskom metodom pretraživanja. Algoritmom pronalazimo duljinu koraka α_k nekom od aproksimativnih metoda spomenutih u ovom radu, dok za vektor smjera p_k zahtijevamo da $\nabla f(x_k)^T p_k < 0$. Utvrdimo konvergenciju općenitog algoritma navedenog u ovom poglavlju. Unutar algoritma želimo izbjeći smjerove $\alpha_k p_k$ koji su skoro ortogonalni na negativni gradijent $-\nabla f(x_k)$. Želimo da kut θ_k između $\alpha_k p_k$ i $-\nabla f(x_k)$ bude udaljen od $\frac{\pi}{2}$, tj. za $\rho > 0$

$$\theta_k \leq \frac{\pi}{2} - \rho, \qquad \forall k. \tag{36}$$

Ako uvjet (36) nije zadovoljen, onda $\nabla f(x_k)^T \alpha_k p_k \to 0$, odnosno narušavamo uvjet $\nabla f(x_k)^T p_k < 0$, koji nam osigurava tvrdnju da se radi o aproksimativnom linijskom pretraživanju.

Algoritam 6 Algoritam minimizacije sa aproksimativnom linijskom metodom pretraživanja

Input: $x_0 \in \mathbb{R}^n$, $\epsilon \in [0, 1)$, $k = 0$, m dovoljno velik
Output: x_k

1: **for** $i = 1$, $i{+}{+}$, $i < m$ **do**
2: **if** $||\nabla f(x_k)|| < \epsilon$ **then**
3: **return** x_k
4: **else**
5: pronađi p_k tako da vrijedi $\nabla f(x_k)^T p_k < 0$
6: **end if**
7: pronađi duljinu koraka α_k koristeći Algoritam pretraživanja unatrag, Goldstein metodu ili (Snažnu) Wolfe- Powell metodu
8: $x_{k+1} \leftarrow x_k + \alpha_k p_k$
9: $k \leftarrow k + 1$
10: **end for**

Teorem 2 ([7]) *Neka $\alpha_k > 0$ u Algoritmu 6 bude izvedena koristeći Algoritam pretraživanja unatrag, Goldsteinovu metodu ili (Snažnu) Wolfe-Powellovu metodu te neka vrijedi (36). Ako ∇f postoji i uniformno je neprekidna na skupu $\{x|f(x) \leq f(x_0)\}$, tada vrijedi točno jedna od sljedećih tvrdnji:*

- $\nabla f(x_k) = 0$, *za neki k*
- $f(x_k) \to \infty$
- $\nabla f(x_k) \to 0$.

Dokaz. Najprije pretpostavimo da $\alpha_k > 0$ zadovoljava Goldsteinovo pravilo te da za svaki k vrijedi $\nabla f(x_k) \neq 0$ (kada $\alpha_k p_k \neq 0$) i da je f omeđena odozdo. Iz uvjeta (26) slijedi da $f(x_k) - f(x_k + \alpha_k p_k) \to 0$, stoga je $-\nabla f(x_k)^T \alpha_k p_k \to 0$. Pretpostavimo također da ne vrijedi $\nabla f(x_k) \to 0$. Tada postoji $\epsilon > 0$ takav da $||\nabla f(x_k)|| > \epsilon$ i $||\alpha p_k|| \to 0$. Kako prema (36) vrijedi $\theta_k \leq \frac{\pi}{2} - \rho$, slijedi

$$\cos(\theta_k) \geq \cos(\frac{\pi}{2} - \rho) = \sin(\rho).$$

Stoga vrijedi

$$-\nabla f(x_k)^T \alpha_k p_k \geq \sin\rho ||\nabla f(x_k)|| ||\alpha_k p_k|| \geq \epsilon \sin\rho ||\alpha_k p_k||.$$

Prema razvoju funkcije f u Taylorov redu imamo:

$$f(x_{k+1}) = f(x_k) + \nabla f(\xi)^T \alpha_k p_k,$$

pri čemu je $\xi \in (x_k, x_{k+1})$. Zbog uniformne konvergencije nam slijedi da $\nabla f(\xi) \to \nabla f(x_k)$ kada $\alpha_k p_k \to 0$

$$f(x_{k+1}) = f(x_k) + \nabla f(x_k)^T \alpha_k p_k + o(||\alpha_k p_k||).$$

Dakle dobivamo,

$$\frac{f(x_k) - f(x_{k+1})}{-\nabla f(x_k)^T \alpha_k p_k} \to 1,$$

što je kontradikcija u odnosu na drugo Goldsteinovo pravilo (35). Zaključujemo da $\nabla f(x_k) \to 0$, čime je dokaz gotov. Analogno dokazujemo za druge metode spomenute u teoremu. □

Literatura

[1] Larry Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific J. Math.*, 16(1):1–3, 1966.

[2] Michael Bartholomew-Biggs. *Nonlinear Optimization with Financial Applications*. Kluver Academic Publishers, 2005.

[3] Doron Levy. *Introduction to Numerical Analysis*. Department of Mathematics and Center for Scientific Computation and Mathematical Modeling (CSCAMM) University of Maryland, 2010.

[4] Jorge J. More, David J. Thuente, and Preprint Mcs-p. Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Software*, 20:286–307, 1992.

[5] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization, second edition*. World Scientific, 2006.

[6] Rudolf Scitovski. *Numerička matematika*. Odjel za matematiku Sveučilišta u Osijeku, 2004.

[7] Wenyu Sun Ya-Xiang Yuan. *Optimization Theory and Methods, Nonlinear Programming*. Springer Science and Business Media, 2006.

[8] Ninoslav Truhar Zoran Tomljanović, Rudolf Scitovski. *Metode optimizacije*. Sveučilište Josipa Jurja Strossmayera u Osijeku - Odjel za matematiku, 2014.

Sažetak:

U radu su opisane iterativne metode bezuvjetne višedimenzionalne optimizacije, točnije metode linijskog pretraživanja. Prije spomenute metode podijeljene su na: egzaktne ili optimalne metode linijskog pretraživanja i aproksimativne ili približne metode linijskog pretraživanja. Unutar Poglavlja 4 opisane su metode optimalnog linijskog pretraživanja uz pripadajuće algoritme i primjere. Aproksimativne metode linijskog pretraživanja uz pripadajuće primjere i algoritme opisane su unutar Poglavlja 5.

Ključne riječi: višedimenzionalna optimizacija, metode linijskog pretraživanja, egzaktno linijsko pretraživanje, optimalno linijsko pretraživanje

Abstract: The paper describes the iterative methods for multidimensional unconstrained optimization, more precisely line search methods. The line search methods are divided into: exact line search methods and approximate line search methods. Within Chapter 4, the methods of optimal line search have been described with the associated algorithms and examples. The aproximate methods of line search with the corresponding examples and algorithms have been described in Chapter 5.

Key words: multidimensional optimization, line search metods, exact line search, approximate line search

Životopis:

Rođena sam 23. svibnja 1992. godine u Zagrebu. 2006. godine završila sam Osnovnu školu August Harambašić u Donjem Miholjcu. Iste godine upisala sam III. Gimnaziju Osijek. Tijekom osnovne i srednje škole sudjelovala sam na županijskim i reginalnim natjecanjima iz više predmeta uključujući i matematiku. Preddiplomski studij matematike na Odjelu za matematiku Sveučilišta J.J. Strossmayera u Osijeku upisala sam 2011. godine. Preddiplomski studij završila sam 2014. godine, te sam upisala diplomski studij matematike na Odjelu za matematiku Sveučilišta J.J. Strossmayera u Osijeku, smjer Matematika i računarstvo.