UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
ODSJEK ZA RAČUNARSTVO I INFORMATIKU

POWELLOV METOD KONJUGIRANIH PRAVACA ZA MINIMIZACIJU FUNKCIJA VIŠE NEZAVISNIH PROMJENLJIVIH

SEMINARSKI RAD IZ NUMERIČKIH ALGORITAMA

Studentice:
Kurtović Esma
Jozić Ivona

Predmetni profesor:
red. prof. dr Jurić Željko

Sarajevo,
februar 2024.

SAŽETAK

Problem optimizacije, bila to minimizacija ili maksimizacija, se svrstava među probleme s najširom primjenom u realnom svijetu. S obzirom da se mnogi problemi mogu modelirati različitim funkcijama, ne čudi da su za njegovo rješavanje poznati brojni metodi i algoritmi. Kako ne postoji metod optimizacije koji je moguće koristi univerzalno u svim situacijama, izbor ovisi o više faktora, kao što su karakteristike konkretnog problema, dostupni podaci i resursi.

U ovom radu, fokus je usmjeren na Powellov metod konjugiranih pravaca za minimizaciju funkcija više nezavisnih promjenljivih, te njegove primjene u realnom svijetu. Data je teorijska pozadina iza samog metoda, kao i osvrt na njegove modifikacije. Implementacija Powellovog algoritma je urađena u programskom jeziku Julia, te je uvršteno nekoliko primjera koji oslikavaju njegovu konkretnu upotrebu. Također, dat je i kratak osvrt na probleme za koje ovaj metod ne radi optimalno.

ABSTRACT

Function optimization, whether it be minimization or maximization, is among the most widely relevant problems in the real world. Considering that many problems can be represented using various functions, it is no surprise that numerous methods and algorithms have been developed regarding this issue. Since there is no universal optimization method, the choice depends on multiple factors such as the problem characteristics, available information, and resources.

This paper focuses on Powell's conjugate directions method for minimizing functions of multiple independent variables, as well as its practical applications. It provides a theoretical background behind the method, along with an overview of its modifications. The implementation of Powell's algorithm is shown in the Julia programming language, including several examples that illustrate its specific use. Additionally, a brief analysis is given about different situations where this method may not be the optimal choice.

SADRŽAJ

1. Teoretski uvod

Kada je u pitanju višedimenzionalna minimizacija, postoje tri glavne tehnike: metodi direktnog pretraživanja, metodi pretraživanja po pravcima i metodi regiona od povjerenja, još poznati i kao metodi ograničenog koraka. Powellov algoritam spada u metode pretraživanja po pravcima, stoga se na preostala dva tipa nećemo detaljnije osvrtati.

Ideja metoda pretraživanja po pravcima je da se krenuvši od neke početne tačke bira pravac (tipično silazni) po kojem će se vršiti minimizacija. Dolaskom do minimuma duž tog pravca, iz trenutne tačke se bira novi i vrši isti slijed koraka. Ovaj postupak se ponavlja sve dok se ne pronađe minimum. Zadatak svih metoda pretraživanja po pravcima je zapravo dati upute kako generisati, odnosno ažurirati postojeći skup pravaca po kojima će se potraga za minimumom nastaviti.

Najjednostavniji metod iz ove skupine, ali koji dokazano često radi vrlo loše, je pretraživanje po koordinatnim osama. Naime, on se susreće s velikim problemom ukoliko grafik funkcije u prostoru ima "duguljastu usku dolinu" u kojoj se nalazi minimum, jer je tada potrebno izuzetno mnogo sitnih koraka za dostizanje istog. Stoga sve naprednije metode pretraživanja po pravcima pokušavaju naći skup koji sadrži ili "dobre" pravce koji nemaju problem s "duguljastim uskim dolinama", ili određeni broj pravaca takvih da minimizacija duž jednog nije pokvarena budućom minimizacijom duž nekog drugog pravca. U drugi navedeni koncept se uklapa ideja konjugiranih pravaca koje koristi Powellov algoritam.

Razmotrimo ideju konjugiranih pravaca. Koristeći razvoj u Taylorov red, proizvoljnu funkciju možemo aproksimirati kao

$$f(\boldsymbol{x}) = f(\boldsymbol{P}) + \sum_i \frac{\partial f}{\partial x_i} x_i + \frac{1}{2} \sum_{i,j} \frac{\partial^2 f}{\partial x_i \partial x_j} x_i x_j + \cdots \approx c - \boldsymbol{b}^T \boldsymbol{x} + \frac{1}{2} \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \quad (*)$$

gdje je $\boldsymbol{P}$ centar koordinatnog sistema, a

$$c \equiv f(\boldsymbol{P}), \qquad \boldsymbol{b} \equiv -\nabla f|_{\boldsymbol{P}}, \qquad [\boldsymbol{A}]_{i,j} \equiv \frac{\partial^2 f}{\partial x_i \partial x_j} \Big|_{\boldsymbol{P}}$$

Primijetimo da je $\boldsymbol{A}$ zapravo Hesseova matrica (Hessian), koja je simetrična ako je zadovoljen Schwarzov teorem, a u tački minimuma i njenoj bliskoj okolici mora biti pozitivno definitna.

Iz ove aproksimacije gradijent dobijamo kao

$$\nabla f = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}.$$

Kretanjem duž nekog pravca, gradijent se mijenja po formuli

$$\delta(\nabla f) = \boldsymbol{A}(\delta \boldsymbol{x})$$

Imajući na umu da je cilj iskoristiti postojeće informacije, kao i činjenicu da, ukoliko minimiziramo funkciju duž nekog pravca $\boldsymbol{u}$, gradijent mora biti okomit na $\boldsymbol{u}$ u tački minimuma, iz te tačke novi pravac $\boldsymbol{v}$ biramo tako da gradijent, odnosno vektor promjene gradijenta, bude okomit na $\boldsymbol{u}$. Dobijamo:

$$0 = \boldsymbol{u} \cdot \delta(\nabla f) = \boldsymbol{u}^T \boldsymbol{A} \boldsymbol{v} \quad (**)$$

Sada konačno dolazimo do pojma konjugiranih pravaca - upravo za vektore (pravce određene vektorima) $\boldsymbol{u}$ i $\boldsymbol{v}$ s ovakvim svojstvima kažemo da su konjugirani (u odnosu na matricu $\boldsymbol{A}$). Kada ova relacija vrijedi za svaki par vektora iz nekog skupa, kažemo da oni čine konjugiran skup. Ukoliko se minimizacija pretraživanjem po pravcima vrši duž pravaca iz konjugiranog skupa, onda nema potrebe za vraćanjem na bilo koji od već pređenih pravaca.

Konačni cilj čitavog postupka je dobiti skup n linearno nezavisnih, međusobno konjugiranih pravaca. Linearna nezavisnost je svakako potrebna kako bi ovaj skup razapinjao prostor $\mathbb{R}^n$, jer minimum tražimo na čitavom prostoru.

$\boldsymbol{th}$. Konjugiran skup od n pravaca u $\mathbb{R}^n$ čini bazu tog prostora, odnosno razapinje $\mathbb{R}^n$.

$\boldsymbol{dokaz}$:

Neka su $\boldsymbol{u}_i,\ i = 1, \dots, n,$ međusobno konjugirani pravci u odnosu na simetričnu, pozitivno definitnu matricu $\boldsymbol{A} \in \mathbb{R}^{n \times n}$. Posmatrajmo linearnu kombinaciju jednaku 0:

$$\sum_{i=1}^{n} \alpha_i \boldsymbol{u}_i = 0,$$

koja nakon množenja s $\boldsymbol{A}$ slijeva postaje

$$\boldsymbol{A} \sum_{i=1}^{n} \alpha_i \boldsymbol{u}_i = \sum_{i=1}^{n} \alpha_i \boldsymbol{A} \boldsymbol{u}_i = 0.$$

Uzevši skalarni proizvod s $\boldsymbol{u}_k$ i koristeći svojstvo konjugiranih vektora $\boldsymbol{u}^T \boldsymbol{A} \boldsymbol{v} = 0$, imamo:

$$\boldsymbol{u}_k \cdot \sum_{i=1}^{n} \alpha_i \boldsymbol{A} \boldsymbol{u}_i = \sum_{i=1}^{n} \alpha_i \boldsymbol{u}_k{}^T \boldsymbol{A} \boldsymbol{u}_i = \alpha_k \boldsymbol{u}_k{}^T \boldsymbol{A} \boldsymbol{u}_k = 0 \quad (***)$$

Na osnovu činjenice da je $\boldsymbol{A}$ pozitivno definitna matrica, vrijedi

$$\boldsymbol{u}_k{}^T \boldsymbol{A}\boldsymbol{u}_k > 0 \quad \forall \boldsymbol{u}_k, \boldsymbol{u}_k \neq \boldsymbol{0},$$

što znači da je u jednačini $(***)$ $\alpha_k = 0,\ \forall k = 1, \dots, n$. Kako je linearna kombinacija vektora $\boldsymbol{u}_i, i = 1, \dots, n,$ jednaka nuli samo kada su koeficijenti te kombinacije α_i također jednaki nuli, zaključujemo da su ovi vektori linearno nezavisni. Kako se skup sastoji od n takvih vektora, oni čine bazu prostora $\mathbb{R}^n$.

Powellov algoritam upravo proizvodi konjugirani skup od n pravaca. Inicijalni skup čine koordinatne ose, a potom se u svakoj iteraciji generiše po jedan novi konjugirani pravac. Jedna iteracija se sastoji od sljedećih koraka, pri čemu je $\boldsymbol{x}_0$ početna aproksimacija minimuma:

1) Za $i = 1, \dots, n$, izvršiti jednodimenzionalnu minimizaciju funkcije $f(\boldsymbol{x}_{i-1} + h\boldsymbol{u}_i)$ po h. Argument za koji ova funkcija dostiže minimum označavamo s h i definišemo novu tačku $\boldsymbol{x}_i = \boldsymbol{x}_{i-1} + h\boldsymbol{u}_i$.
2) Za $i = 1, \dots, n-1$, zamijeniti $\boldsymbol{u}_i$ s $\boldsymbol{u}_{i+1}$.
3) Zamijeniti $\boldsymbol{u}_n$ s $\boldsymbol{x}_n - \boldsymbol{x}_0$.
4) Izvršiti jednodimenzionalnu minimizaciju $f(\boldsymbol{x}_n + h\boldsymbol{u}_n)$ i zamijeniti $\boldsymbol{x}_0$ s $\boldsymbol{x}_n + h\boldsymbol{u}_n$, pri čemu je h ponovo argument za koji se dostiže taj minimum.

Za upotpunjavanje algoritma će biti naveden i dokaz teoreme na kojoj se temelji korak 3).

th. Ako funkcija $f(\boldsymbol{x})$ duž nekog pravca $\boldsymbol{u}$, polazeći od tačke $\boldsymbol{x_i}^*$, dostiže minimum u tački $\boldsymbol{x_i}$, za $i = 0{,}1$, onda je pravac $\boldsymbol{x}_1 - \boldsymbol{x}_0$ konjugiran pravcu $\boldsymbol{u}$.

dokaz:

Na osnovu uvjeta koji vrijede za tačke $\boldsymbol{x_i}$, $i = 0{,}1$, slijedi da je

$$\frac{\partial}{\partial h} f(\boldsymbol{x}_i + h\boldsymbol{u}) = 0$$

za $h = 0$. Stoga, iz $(*)$:

$$\boldsymbol{u}^T(\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{b}) = 0. \quad (****)$$

Oduzimanjem jednačina $(****)$ za $i = 0{,}1$ dobiva se

$$\boldsymbol{u}^T \boldsymbol{A}(\boldsymbol{x}_1 - \boldsymbol{x}_0) = 0,$$

pa iz $(**)$ slijedi da su pravci $\boldsymbol{u}$ i $(\boldsymbol{x}_1 - \boldsymbol{x}_0)$ konjugirani.

Ako je f kvadratna funkcija, ovaj postupak će poslije k iteracija dati skup pravaca $\boldsymbol{u}_i$ čijih će zadnjih k elemenata ($i = n-k,\ n-k+1, \dots, n$) biti međusobno konjugirani. Samim tim, n iteracija egzaktno minimizira kvadratnu funkciju. Za slučaj kada funkcija koja se minimizira nije kvadratna, postupak treba nastaviti dok se ne ispuni neki od uobičajenih kriterija za prekid. Pritom, ukoliko se funkcija u okolini minimuma ponaša približno kao kvadratna, i sama konvergencija je kvadratna - rješenju se primičemo vrlo brzo nakon što smo mu došli dovoljno blizu.

Problem s Powellovim algoritmom je što ponekad, nakon većeg broja iteracija, vektori $\boldsymbol{u}_i$ postaju linearno zavisni. Time pretraga postaje ograničena samo na neki podskup $\mathbb{R}^n$, a prethodno je naglašen značaj toga da korišteni skup pravaca (vektora) razapinje čitav prostor. Postoji više načina kako se može riješiti ovaj problem:

i) Nakon svakih n ili $n+1$ iteracija se skup pravaca resetuje na početni, odnosno na koordinatne ose (kolone jedinične matrice). Međutim, ovim se periodično odbacuje nezanemarljiv dio dotada stečenih informacija o funkciji.

ii) Brentova modifikacija: Skup pravaca se može periodično resetovati na kolone bilo koje ortogonalne matrice $\boldsymbol{Q}$ (dakle, to ne mora biti jedinična matrica). Matrica $\boldsymbol{Q}$ se bira tako da skup vektora ostane konjugiran ako je f kvadratna funkcija. Kako su novi pravci pretrage ortogonalni, riješen je problem obuhvatanja čitavog prostora $\mathbb{R}^n$. Ovaj algoritam je nešto kompleksniji, budući da zahtijeva upotrebu SVD algoritma, no vjerovatno je trenutno i najbolja poznata modifikacija.

iii) Powellova modifikacija: Na osnovu koraka 2) i 3) gore navedenog algoritma, vidimo se da u svakoj iteraciji odbacuje pravac $\boldsymbol{u}_1$ a čuvaju ostali pravci, pri čemu se na kraju u skup dodaje pravac $\boldsymbol{x}_n - \boldsymbol{x}_0$. U ovoj modifikaciji se na isti način vrši dodavanje, ali se pritom ne odbacuje svaki put $\boldsymbol{u}_1$, već onaj pravac duž kojeg se funkcija najviše smanjila – vjerovatno je da je on važna komponenta novododanog pravca. Ovime se smanjuje vjerovatnoća stvaranja linearne zavisnosti. Postoje određeni izuzeci u slučaju kojih se zadržava stari skup pravaca.

2. Implementacija algoritma u programskom jeziku Julia

```
using LinearAlgebra

function min_bracket(f, x0, hinit = 1e-5, hmax = 1e10, lambda = 1.4)
    h = hinit
    a = 0; b = 0
    f0 = 0; f1 = 0; f2 = 0

    if f(x0) > f(x0 + h) || f(x0) > f(x0 - h)
        while h < hmax
            a = x0 - h
            b = x0 + h
            f1 = f(a)
            f2 = f(b)
            f0 = f(x0)

            while isinf(f1)
                h /= 2 * (1 + lambda)
                a = x0 - h
                f1 = f(a)
            end

            while isinf(f2)
                h /= 2 * (1 + lambda)
                b = x0 + h
                f2 = f(b)
            end

            if f1 > f0 && f2 > f0
                return (a, b, x0)
            end

            h *= lambda

            if f0 > f1
                x0 = a
            elseif f0 > f2
                x0 = b
            end
        end

        throw(DomainError(-2, "Minimum could not be bracketed"))
```

```
    else
        return (x0 - h, x0 + h, x0)
    end
end

function golden_ratio_min(f, a, b, c, eps = 1e-8)
    phi = (1 + sqrt(5)) / 2
    d = 0

    if abs(c - a) < abs(b - c)
        d = b - (b - c) / phi
    else
        d = c
        c = a + (c - a) / phi
    end

    u = f(c)
    v = f(d)

    while abs(b - a) > eps
        if u < v
            b = d
            d = c
            c = a + (c - a) / phi
            v = u
            u = f(c)
        else
            a = c
            c = d
            d = b - (b - d) / phi
            u = v
            v = f(d)
        end
    end

    return (a + b) / 2
end

function find_minimum(f)
    (a, b, c) = (0.0, 0.0, 0.0)
    try
        (a, b, c) = min_bracket(f, 0)
    catch
        throw(DomainError(-3, "Minimum could not be bracketed"))
    end
```

```
    return golden_ratio_min(f, a, b, c)
end

function powell_min(f, x0, points, maxiter = 100, eps = 1e-8)
    n = length(x0)
    x = zeros(n, n)
    u = I(n) + zeros(n, n)
    h = 0

    for k in 1 : maxiter
        push!(points, x0)

        try
            h = find_minimum((h) -> f(x0 + h * u[:, 1]))
        catch
            break
        end

        x[:, 1] = x0 + h * u[:, 1]

        for i in 2 : n
            try
                h = find_minimum((h) -> f(x[:, i - 1] + h * u[:, i]))
            catch
                break
            end

            x[:, i] = x[:, i - 1] + h * u[:, i]
        end

        for i in 1 : n - 1
            u[:, i] = u[:, i + 1]
        end

        u[:, n] = x[:, n] - x0
        if norm(u[:, n]) < eps
            return x0
        end

        try
            h = find_minimum((h) -> f(x[:, n] + h * u[:, n]))
        catch
            break
        end
```

```
            x0 = x[:, n] + h * u[:, n]
        end

        throw(DomainError(-1, "Minimum has not been found"))
    end
```

Glavna funkcija, koja zapravo i vrši minimizaciju funkcije više promjenjivih, jeste `powell_min`. Kao parametre prima funkciju koja se minimizira `f`, početnu aproksimaciju minimuma `x0`, vektor `points` u koji će se spremati tačke `x0` sve dok minimum ne bude pronađen, te maksimalan broj iteracija `maxiter` i toleranciju `eps`. Unutar petlje `for k in 1 : maxiter` se vrše pojedinačne iteracije na način kako je opisano u teoretskom dijelu, pri čemu oznake odgovaraju gore navedenom.

Za jednodimenzionalnu minimizaciju po `h` se koristi pomoćna funkcija `find_minimum`, koja kao parametar prima samo funkciju `f`. Unutar nje se poziva `min_bracket`, čiji se rezultati prosljeđuju u `golden_ratio_min`. `min_bracket` služi za ogradu minimuma funkcije `f` krenuvši od početne tačke `x0`, a parametri su početni i maksimalni korak `hinit` i `hmax`, te faktor širenja koraka `lambda`. Povratna vrijednost je trojka `(a, b, x0)`, koja predstavlja traženu ogradu minimuma, odnosno za koju vrijedi `a < x0 < b`, `f(a) > f(x0)` i `f(b) > f(x0)`. Sam algoritam funkcioniše tako da se na osnovu odnosa vrijednosti funkcije u ove tri tačke pomjera središnja tačka `x0`, a zatim se provjerava da li ona, skupa s tačkama `a` i `b` na udaljenosti `h`, čini ogradu. Petlje `while isinf(f1)` i `while isinf(f2)` služe za provjeru da li je došlo do izlaska iz domena, pa se u tom slučaju vrši korekcija koraka.

`golden_ratio_min` vrši minimizaciju funkcije jedne promjenjive koristeći algoritam zlatnog presjeka, i to verziju bez ograničenja na unimodalnost. Kao parametre prima funkciju `f`, ogradu minimuma `(a, c, b)` i toleranciju `eps`. Radi tako da pomjeramo granice intervala `(a, b)` na osnovu izračunatih vrijednosti, sve dok dužina tog intervala ne postane manja od tolerancije `eps`, odnosno željene tačnosti. Na kraju se kao lokacija minimuma vraća središte intervala.

Funkcije `min_bracket` i `find_minimum` bacaju izuzetak ukoliko minimum ne uspije biti ograđen, a funkcija `powell_min` ukoliko ne uspije pronaći minimum ni nakon obavljenog maksimalnog broja iteracija. Algoritam `powell_min` terminira ukoliko je norma vektora koji određuje novododani pravac `un`, a koji je jednak `xn - x0`, manja od zadane tolerancije `eps`.

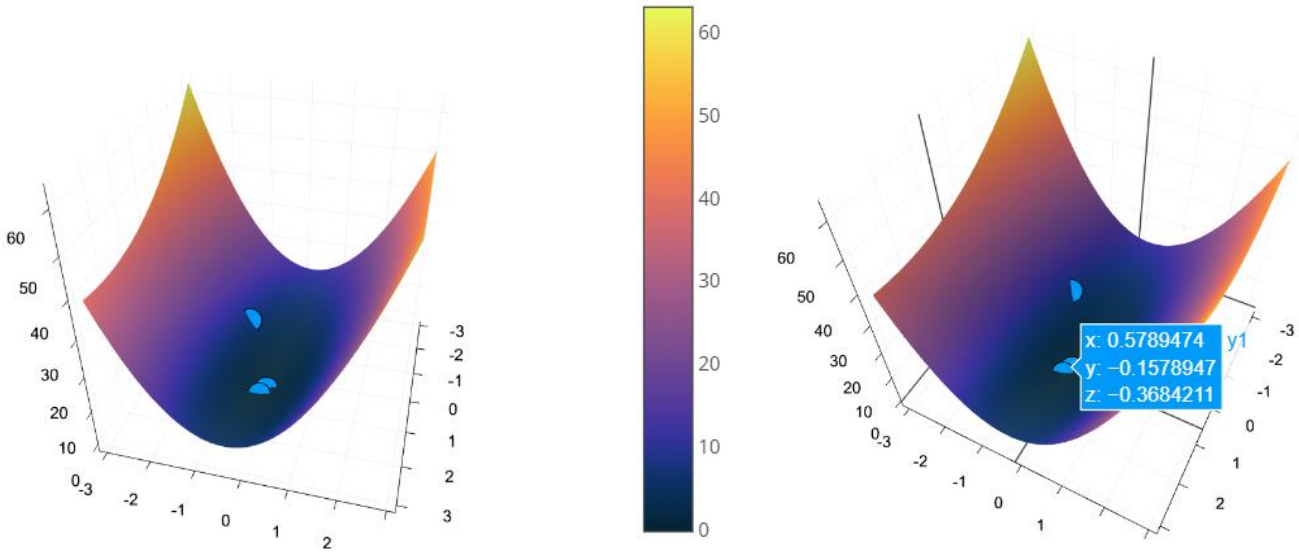
Prikazani kod ćemo demonstrirati za pronalazak minimuma nekoliko karakterističnih funkcija za koje Powellov metod radi dobro.

Na prvom mjestu je obična kvadratna funkcija dvije nezavisne varijable

$$f(x, y) = x^2 + 5y^2 + xy - x + y.$$

Početna tačka	Tačka minimuma	Vrijednost minimuma	Tačke x0[1]
[-1.0, -1.0]	[0.5789473618567542, -0.1578947364333902]	-0.368421052631579	[-1.0, -1.0] [0.81818, -0.27273] [0.5789473618567542, -0.1578947364333902]

Tabela 1. Podaci o minimizaciji za kvadratnu funkciju



Slika 1. Kvadratna funkcija – grafik i nađena tačka minimuma

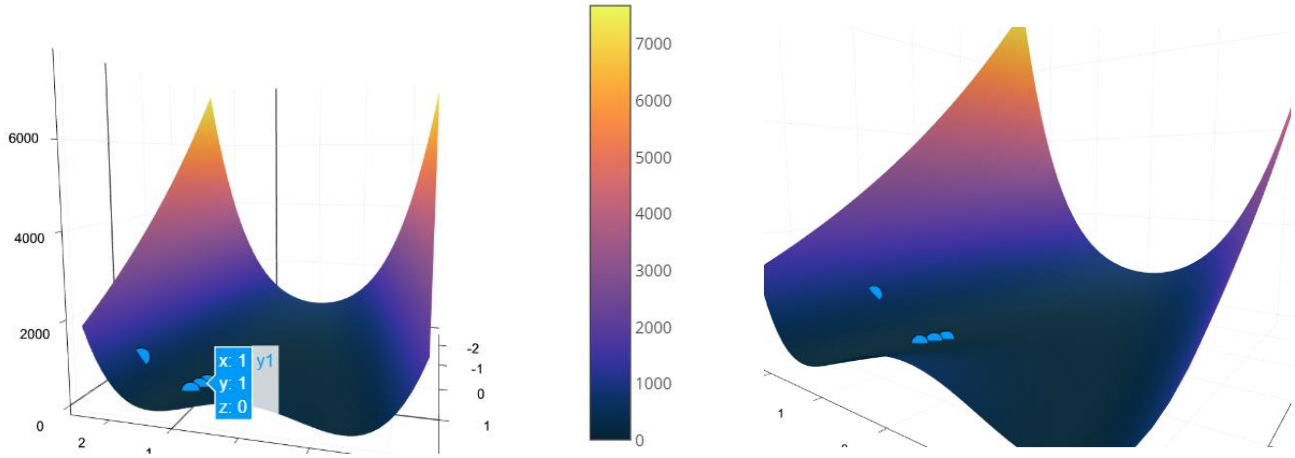
Sljedeća karakteristična funkcija je Rosenbrockova funkcija, poznata još i kao Banana funkcija. Oblik na koji ćemo primijeniti implementirani algoritam je:

$$f(x,y) = (1-x)^2 + 100(y-x^2)^2.$$

Početna tačka	Tačka minimuma	Vrijednost minimuma	Tačke x0
[2.0, 1.3]	[1.0000000000004725, 1.0000000000004294]	2.680632107449956e-23	[2.0, 1.3] [1.13964, 1.29939] [1.06853, 1.13705] [1.00092, 1.00099] [1.00003, 1.00006] [1.0000000000004725, 1.0000000000004294]

Tabela 2. Podaci o minimizaciji za Rosenbrockovu funkciju

[1] Međuvrijednosti prije tačke minimuma su zaokružene na 5 decimala radi jednostavnijeg prikaza.



Slika 2. Rosenbrockova funkcija – grafik i nađena tačka minimuma

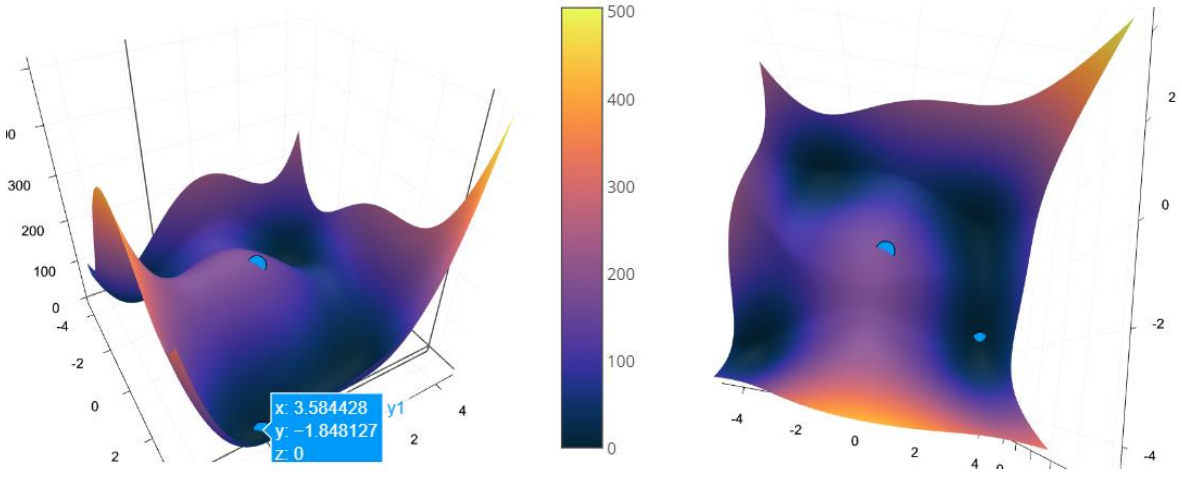
Powellov metod primijenili smo na još nekoliko funkcija sa sljedećim oblicima:

- Himmelblauova funkcija

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$$

Početna tačka	Tačka minimuma	Vrijednost minimuma	Tačke x0
[0.0, 0.0]	[3.5844283403372863, -1.8481265268913947]	8.401800275314398e-20	[0.0, 0.0] [3.57954, -1.90081] [3.58444, -1.84813] [3.5844283403372863, -1.8481265268913947]

Tabela 3. Podaci o minimizaciji za Himmelblauovu funkciju



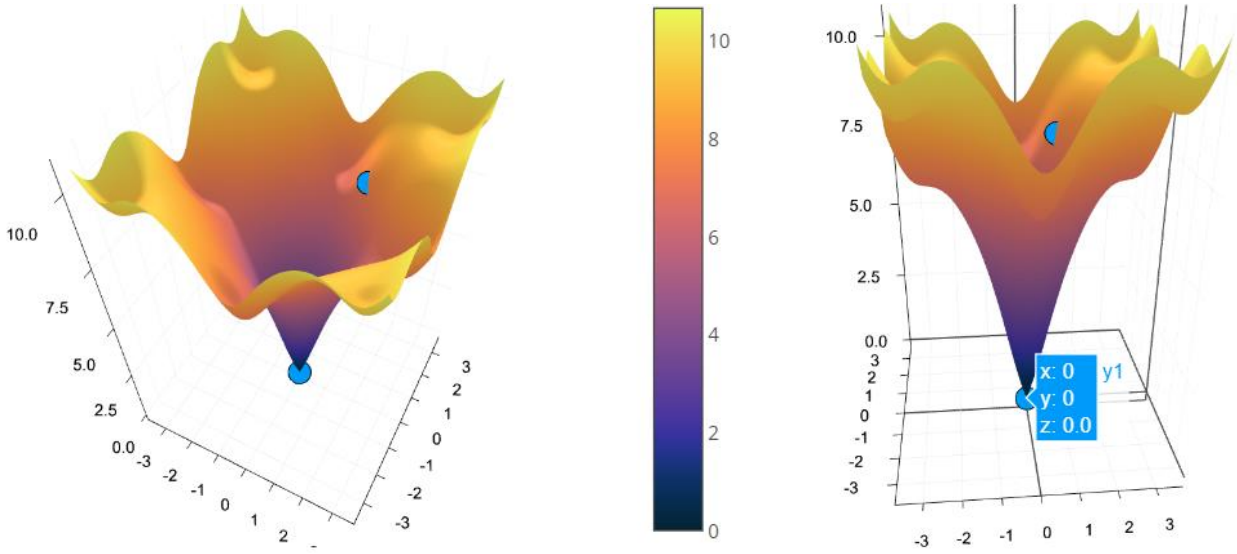
Slika 3. Himmelblauova funkcija – grafik i nađena tačka minimuma

- Ackleyeva funkcija

$$f(x,y) = -20e^{-0.2\sqrt{\frac{x^2+y^2}{2}}} - e^{\frac{1}{2}(\cos(2x)+\cos(2y))} - 20 + e$$

Početna tačka	Tačka minimuma	Vrijednost minimuma	Tačke x0
[1.0, 2.0]	[3.881300633112983e-8, -1.773196030454731e-8]	1.206937274567110e-7	[1.0, 2.0] [3.881300633112983e-8, -1.773196030454731e-8]

Tabela 4. Podaci o minimizaciji za Ackleyevu funkciju



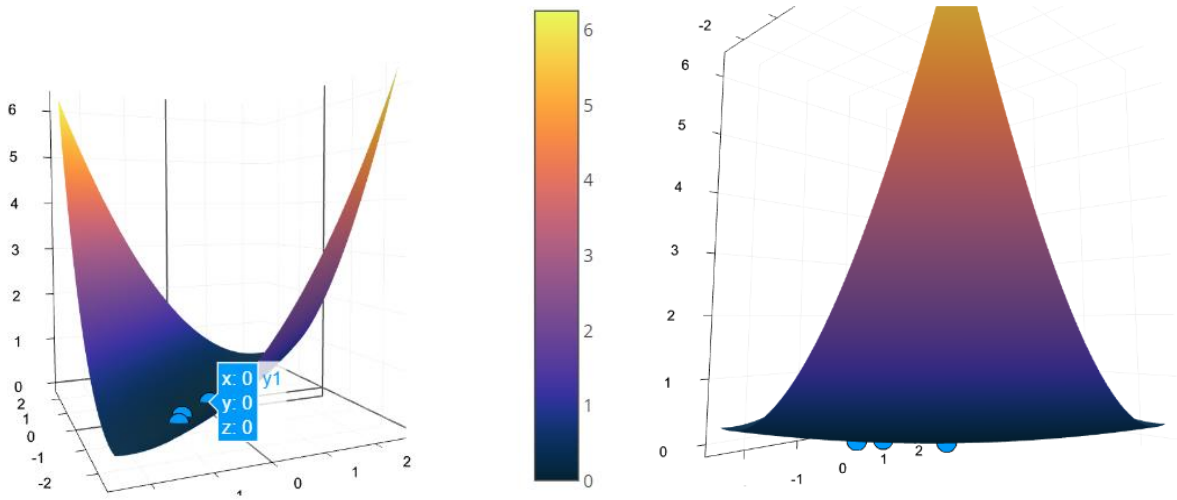
Slika 4. Ackleyeva funkcija – grafik i nađena tačka minimuma

- Matyaseva funkcija

$$f(x,y) = 0.26(x^2 + y^2) - 0.48xy$$

Početna tačka	Tačka minimuma	Vrijednost minimuma	Tačke x0
[-1.0, -1.0]	[-3.836552187053144e-9, -7.575456706376826e-9]	4.7971912209920124e-18	[-1.0, -1.0] [-0.80412, -0.62331] [-3.836552187053144e-9, -7.575456706376826e-9]

Tabela 5. Podaci o minimizaciji za Matyasevu funkciju



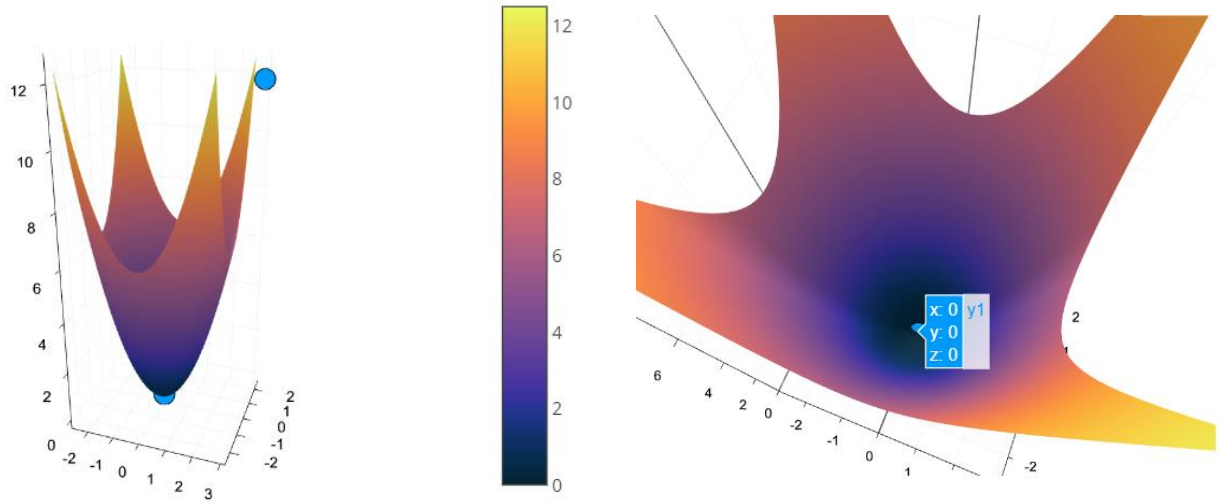
Slika 5. Matyaseva funkcija – grafik i nađena tačka minimuma

- Sferna funkcija

$$f(x, y) = x^2 + y^2$$

Početna tačka	Tačka minimuma	Vrijednost minimuma	Tačke x0
[3.0, 1.7]	[3.6880316574786663e-9, -8.464173504715402e-9]	8.524381062449106e-17	[3.0, 1.7] [3.6880316574786663e-9, -8.464173504715402e-9]

Tabela 6. Podaci o minimizaciji za sfernu funkciju



Slika 6. Sferna funkcija – grafik i nađena tačka minimuma

- Primjer funkcije više od 2 varijable:

$$f(x_1, x_2, x_3, x_4, x_5) = 2\sin({x_1}^2 + 2{x_2}^2 + 10{x_3}^2 + {x_4}^2 + 12{x_5}^2 - 5) + 3$$

Početna tačka	Tačka minimuma	Vrijednost minimuma	Tačke `x0`
`[1.0, 2.0, 0.3, 3.3, 1.2]`	`[2.0145261273271005, 2.0, 0.3, 3.3, 1.2]`	`1.0`	`[1.0, 2.0, 0.3, 3.3, 1.2]` `[2.0145261273271005, 2.0, 0.3, 3.3, 1.2]`

Tabela 7. Podaci o minimizaciji za proizvoljnu funkciju više od 2 varijable

Analizirajući broj tačaka `x0` i očekivani broj iteracija koji je za kvadratne funkcije n, a za ostale funkcije može biti i nešto veći, gdje n predstavlja dimenzionalnost prostora, možemo zaključiti da Powellov metod radi u skladu s očekivanjima. Već na prvom primjeru vidimo da je obična kvadratna funkcija minimizirana u n iteracija. Primjetno je da broj iteracija obično nije velik ni za ostale funkcije na kojima je algoritam testiran, pa tako npr. Rosenbrockovu funkciju, s kojom obično pretraživanje po koordinatnim osama ima priličan problem, uspijeva minimizirati u svega nekoliko iteracija.

3. Primjene algoritma u praksi

Pored konkretnih primjera primjene Powellovog metoda datih u prethodnom odjeljku, značajno je izdvojiti još neke oblasti i situacije u kojima se može primjenjivati i daje veoma dobre rezultate.

Na prvom mjestu se koristi pri nalaženju minimuma funkcija koje nisu diferencijabilne, zatim pri podešavanju parametara u mašinskom učenju, registraciji slike, molekularnom modeliranju u hemiji, procesiranju signala, te portfolio optimizaciji. U nastavku slijede primjeri koji oslikavaju neke od spomenutih primjena.

Optimizacija u mašinskom učenju podrazumijeva pronalaženje najboljeg skupa parametara koji će minimizirati trošak ili funkciju gubitka, a sve u cilju maksimizacije izvedbe i tačnosti modela. Finim podešavanjem parametara modela kroz optimizaciju, modeli mašinskog učenja mogu napraviti bolja predviđanja i učinkovito riješiti širok raspon zadataka u različitim oblastima. Kako je optimizaciju potrebno vršiti više puta tokom trajanja procesa, postoji nekoliko različitih načina kako to izvesti, a među njima se nalazi i metod konjugiranih pravaca.

Registracija slike je jedan od zadataka sa kojima se susrećemo prilikom obrade fotografija, a podrazumijeva usklađivanje dvije ili više fotografija dobivenih u različitim momentima, različitim uvjetima fotografisanja ili fotografisanih iz različitih perspektiva. Razlike koje se javljaju zbog navedenih razloga je potrebno što je god moguće više minimizirati, pri čemu se minimizacija vrši prilikom dodavanja nove fotografije, a tehnika koja se može primijeniti je Powellov metod konjugiranih pravaca.

Kao što je ranije spomenuto, ovaj metod se može koristi i prilikom molekularnog modeliranja u hemiji, konkretno za optimizaciju molekularne geometrije (dužine veza, uglovi) u simulacijama kvantne hemije, ali isto tako i za analizu EPR spektra molekule. EPR spektar je tehnika proučavanja molekula s neuparenim elektronima.

Powellov metod svoju primjenu nalazi i u finansijama, odnosno portfolio optimizaciji, za maksimizaciju prihoda, te minimizaciju rizika gubitka povezanih s uloženom svotom. Pretpostavimo da imamo skup dionica s očekivanim dobitkom i vrijednostima rizika. Ovim dionicama dodjeljujemo težine kako bismo maksimizirali dobitak uz smanjenje rizika. Powellov algoritam se koristi za optimizaciju težina, a funkcija cilja uključuje očekivane dobiti i rizike.

Za kraj se još osvrnimo na neke primjene i primjere u kojima Powellov metod konjugiranih pravaca i nije baš najsretniji izbor za minimizaciju.

Kako je Powellov metod jedan od metoda pretrage po pravcima, ukoliko se radi o prevelikoj dimenzionalnosti, odnosno funkciji s velikim brojem nezavisnih promenljivih, stalno ažuriranje pravaca postaje preskupo te bi u tom slučaju značajno bolje rezultate dao gradijentni metod.

Slična situacija se javlja i s loše uvjetovanim funkcijama, koje posjeduje regije na kojima je gradijent vrlo ravan ili vrlo strm, ili ukoliko funkcija nije glatka, pa dolazi do spore konvergencije prilikom upotrebe Powellovog metoda. Kako bi se konvergencija ubrzala, bolje rješenje bi bila upotreba gradijentnog metoda.

4. Zaključak i diskusija

Za početak ćemo se osvrnuti na činjenicu da ovaj algoritam, kao i drugi, pronalazi lokalne minimume funkcije. Kako se kao jedan od ulaznih parametara zadaje i tačka koja predstavlja početnu aproksimaciju minimuma, tako i od te početne tačke zavisi koji će lokalni minimum biti pronađen. Upravo s ovim smo se susreli prilikom testiranja implementacije algoritma, konkretno na primjeru funkcije $2x^3 + xy^3 - 10xy + y^2$. Kako ova funkcija na $\mathbb{R}^2$ zapravo uopće nije ograničena odozdo, pokazuje se da čak i prilično mala razlika u izboru početne tačke utječe na to da li će lokalni minimum biti pronađen, ili će potraga biti neuspješna i doći do bacanja izuzetka.

Još jedna od manjkavosti metoda koja je uočena u toku njegovog testiranja u programskom jeziku Julia je nailazak na sedlastu tačku funkcije. Naime, ponašanje metoda u tom slučaju je nepredvidivo, s obzirom da postoje slučajevi u kojima prepoznaje da je riječ o sedlastoj tački i uspješno se izvlači iz nje, dok postoje i oni u kojima umjesto sedla prepoznaje lokalni minimum.

Pored toga, gledajući ponuđenu implementaciju, kao podalgoritam za jednodimenzionalnu minimizaciju je korišten nešto jednostavniji algoritam zlatnog presjeka, što opet ostavlja prostora za potencijalne probleme. Kako se Powellov metod može primijeniti na širok spektar funkcija, vjerovatno bi jedno od mogućih unapređenja bilo implementirati veći broj različitih podalgoritama za jednodimenzionalnu minimizaciju koji bi se primjenjivali u raznim nestandardnim situacijama.

Što se Powellovog metoda općenito tiče, rečeno je da spada u metode za minimizaciju funkcije korištenjem pretraživanja po pravcima. U ovom radu je data teorijska pozadina samog metoda kao i njegova implementacija čije smo potencijalne nedostatke prethodno istaknuli. Ono što je posebno značajno istaknuti je široka primjena metoda u različitim oblastima za optimizaciju kako resursa tako i cijelog procesa. Navedene su i situacije u kojima bi se umjesto ovog metoda bilo bolje odlučiti za neki drugi, kao što je gradijentni metod. Kako metod ima dobru bazu, na osnovu njega su razvijene i neke modifikacije, detaljnije opisane u prvom poglavlju, koje omogućavaju njegovu primjenu čak i u situacijama u kojima izvorni – ovdje obrađeni metod, ne daje najbolje rezultate.

Smatramo da je ovim radom postignut željeni cilj, što je bilo detaljnije se upoznati s Powellovim metodom za optimizaciju i njegovim mogućnostima, ali i uvidjeti nedostatke ovog i sličnih algoritama koji ostavljaju prostora za napredak i dalje istraživanje.

5. Popis literature

[1] Ž. Jurić, *Numerički algoritmi,* Univerzitet u Sarajevu, Sarajevo, 2018, pp. 377 - 380.

[2] H. Press, A. Teukolsky, T. Vetterling, P. Flannery, *Numerical Recipes*, 3rd ed., Cambridge University Press, New York, 2007, https://e-maxx.ru/bookz/files/numerical_recipes.pdf, pp. 533 - 538.

[3] P. Brent, *Algorithms for minimization without derivatives*, Thomas J. Watson Research Center Yorktown Heights, New York, 1973, https://maths-people.anu.edu.au/~brent/pd/rpb011i.pdf, pp. 68 - 72.

[4] *Conjugate Directions*, https://home.cc.umanitoba.ca/~lovetrij/cECE7670/2005/Slides/slides4.pdf, pp. 10 - 16.

[5] *Multivariable optimization*, https://web.iitd.ac.in/~achawla/public_html/742/multivaroptupd.pdf, [pristupljeno: 16. februar 2024.], pp. 25

[6] Surjanovic S., Bingham D., *Virtual Library of Simulation Experiments: Test Functions and Datasets*, *Optimization Test Problems - Matyas function*, Simon Fraser University, 2013., https://www.sfu.ca/~ssurjano/matya.html, [pristupljeno: 16. februar 2024.]

[7] Surjanovic S., Bingham D., *Virtual Library of Simulation Experiments: Test Functions and Datasets*, *Optimization Test Problems - Ackley function*, Simon Fraser University, 2013., https://www.sfu.ca/~ssurjano/ackley.html, [pristupljeno: 16. februar 2024.]

[8] S.J. Petersen, *Methods of Optimization for Numerical Algorithms*, Bsc dissertation, Dept. of mathematics and applied mathematics, University of Groningen, 2017, https://fse.studenttheses.ub.rug.nl/15741/2/BSC_Math_2017_Petersen_SJ.pdf, pp. 27 - 32.

[9] V. Agarwal, *Optimization in Machine Learning (Hyperparameter Optimization)*, juli 2023, https://medium.com/@craackmouse/optimization-in-machine-learning-hyperparameter-optimization-60347799627d, [pristupljeno: 5. februar 2024.]

[10] F. Wang, *An image registration method based on Powell optimization algorithm*, januar 2017, https://www.researchgate.net/publication/319469884_An_image_registration_method_based_on_Powell_optimization_algorithm, [pristupljeno: 5. februar 2024.]

[11] T. Spałek, P. Pietrzyk, Z. Sojka, *Application of the Genetic Algorithm Joint with the Powell Method to Nonlinear Least-Squares Fitting of Powder EPR Spectra*, novembar 2004, https://pubs.acs.org/doi/abs/10.1021/ci049863s, [pristupljeno: 5. februar 2024.]

6. Popis slika i tabela