Izvještaj - Laboratorijska vježba 3

Simplex - standardni oblik LP uz korištenje AI alata

Bakir Činjarević 19705 & Amar Handanagić 19089

04.11.2025.

1 Uvod

Cilj ove laboratorijske vježbe je razumijevanje i implementacija Simplex metode za rješavanje standardnog oblika problema linearnog programiranja u programskom jeziku Julia. Vježba uključuje korištenje AI alata (Claude) za pomoć u pisanju, testiranju i razumijevanju koda.

Standardni oblik linearnog programiranja glasi:

$$\begin{aligned} \max \quad & Z(x) = c^T x \\ \text{p.o.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{1}$$

2 Prompt korišten za AI alat

Za implementaciju Simplex metode u Juliji korišten je sljedeći prompt:

> *"Implementiraj Simplex metodu u programskom jeziku Julia za rješavanje standardnog oblika problema linearnog programiranja. Funkcija treba da se zove* **`rijesi_simplex(A, b, c)`** *i prima matricu A (koeficijenti ograničenja), vektor b (desne strane), i vektor c (koeficijenti funkcije cilja). Funkcija treba da:*
>
> - *Kreira proširenu Simplex tabelu sa slack varijablama*
> - *Implementira Dantzigovo pravilo pivotiranja (odabir najnegativnijeg koeficijenta)*
> - *Provjerava optimalnost rješenja*
> - *Detektuje beskonačna rješenja (kad su svi elementi u pivot koloni ≤ 0)*
> - *Detektuje degeneraciju*
> - *Ispisuje svaku iteraciju tabele*
> - *Vraća optimalno rješenje x, vrijednost funkcije cilja Z, i status*
>
> *Koristi pravilo pravougaonika za pivot operacije: $a'_{ij} = a_{ij} - \frac{a_{iq} \cdot a_{pj}}{pivot}$"*

3 Generisani kod

Kompletan Julia kod za implementaciju Simplex metode nalazi se u priloženom `.jl` fajlu. Ključni dijelovi implementacije su:

```
# Implementacija Simplex metode za rje avanje standardnog oblik LP problema
# Standardni oblik: max Z(x) = c^T x, p.o. Ax    b, x    0

using Printf
using LinearAlgebra

function rijesi_simplex(A, b, c; verbose=true)
    """
    Rje ava standardni oblik LP problema koriste i Simplex metodu.

    Parametri:
    - A: matrica koeficijenata ograni enja (m   n)
    - b: vektor desnih strana (m   1)
    - c: vektor koeficijenata funkcije cilja (n   1)
    - verbose: da li ispisivati iteracije (default: true)

    Povratna vrijednost:
    - x: optimalno rje enje (vektor)
    - Z: optimalna vrijednost funkcije cilja
    - status: "OPTIMAL", "UNBOUNDED", "DEGENERATE", ili "INFEASIBLE"
    """

    m, n = size(A)

    # Provjera da li su dimenzije kompatibilne
    if length(b) != m
        error("Dimenzije A i b nisu kompatibilne")
    end
    if length(c) != n
        error("Dimenzije A i c nisu kompatibilne")
    end

    # Provjera da li su svi elementi b nenegativni
    if any(b .< 0)
        error("Svi elementi b moraju biti nenegativni za standardni oblik")
    end

    # Kreiranje pro irene tabele (dodavanje slack varijabli)
    tableau = zeros(m + 1, n + m + 1)

    # Kopiranje A u tabelu
    tableau[1:m, 1:n] = A

```

```
    # Dodavanje jedini ne matrice za slack varijable
    tableau[1:m, (n+1):(n+m)] = Matrix{Float64}(I, m, m)

    # Dodavanje b u posljednju kolonu
    tableau[1:m, end] = b

    # Dodavanje funkcije cilja (negativni znak jer je maksimizacija)
    tableau[end, 1:n] = -c
    tableau[end, (n+1):(n+m)] = zeros(m)
    tableau[end, end] = 0.0

    # Lista baznih varijabli (indeksi slack varijabli u po etnoj bazi)
    basic_vars = collect(n+1:n+m)

    iteration = 0

    if verbose
        println("="^80)
        println("PO ETNA SIMPLEX TABELA")
        println("="^80)
        print_tableau(tableau, basic_vars, n, m, iteration)
    end

    max_iterations = 1000  # Za tita od beskona ne petlje

    while iteration < max_iterations
        iteration += 1

        # Provjera optimalnosti
        reduced_costs = tableau[end, 1:(n+m)]

        if all(reduced_costs .>= -1e-10)
            # Optimalno rje enje prona eno
            x = zeros(n)

            # Izdvajanje vrijednosti originalnih varijabli
            for i in 1:m
                if basic_vars[i] <= n
                    x[basic_vars[i]] = tableau[i, end]
                end
            end

            Z = tableau[end, end]

            # Provjera degeneracije
            degenerate = false
            for i in 1:m
                if abs(tableau[i, end]) < 1e-10
                    degenerate = true
```

```
                    break
                end
            end

            if verbose
                println("\n" * "="^80)
                println("OPTIMALNO RJE ENJE PRONA ENO nakon $iteration iteracija")
                if degenerate
                    println("NAPOMENA: Rje enje je DEGENERISANO")
                end
                println("="^80)
            end

            status = degenerate ? "DEGENERATE" : "OPTIMAL"
            return x, Z, status
        end

        # Odre ivanje ulazne varijable (najnegativniji koeficijent)
        entering_idx = argmin(reduced_costs)

        if reduced_costs[entering_idx] >= -1e-10
            continue
        end

        # Provjera da li je problem neograni en
        pivot_col = tableau[1:m, entering_idx]

        if all(pivot_col .<= 1e-10)
            if verbose
                println("\n" * "="^80)
                println("PROBLEM JE NEOGRANI EN - beskona no rje enje")
                println("="^80)
            end
            return zeros(n), Inf, "UNBOUNDED"
        end

        # Odre ivanje izlazne varijable (minimum ratio test)
        ratios = zeros(m)
        for i in 1:m
            if pivot_col[i] > 1e-10
                ratios[i] = tableau[i, end] / pivot_col[i]
            else
                ratios[i] = Inf
            end
        end

        # Provjera za degeneraciju (vi e minimuma)
```

```julia
        min_ratio = minimum(ratios)
        leaving_candidates = findall(x -> abs(x - min_ratio) < 1e-10, ratios)

        leaving_row = leaving_candidates[1]

        # A uriranje baznih varijabli
        basic_vars[leaving_row] = entering_idx

        # Pivot operacija
        pivot_element = tableau[leaving_row, entering_idx]

        # Normalizacija pivot reda
        tableau[leaving_row, :] ./= pivot_element

        # Eliminacija u ostalim redovima
        for i in 1:(m+1)
            if i != leaving_row
                multiplier = tableau[i, entering_idx]
                tableau[i, :] .-= multiplier .* tableau[leaving_row, :]
            end
        end

        if verbose
            println("\n" * "-"^80)
            println("ITERACIJA $iteration")
            println("-"^80)
            println("Ulazna varijabla: x$(entering_idx <= n ? entering_idx : "s$(entering_idx-n)")")
            println("Izlazna varijabla: x$(basic_vars[leaving_row] <= n ? basic_vars[leaving_row] : "s$(basic_vars[leaving_row]-n)")")
            print_tableau(tableau, basic_vars, n, m, iteration)
        end
    end

    if verbose
        println("\nMaksimalni broj iteracija dostignut!")
    end

    return zeros(n), 0.0, "MAX_ITERATIONS"
end
```

Listing 1: Glavna funkcija rijesi_simplex

4 Test primjeri

4.1 Test primjer 1: Standardni LP problem

Formulacija problema:

$$\begin{aligned} \max \quad & Z = 3x_1 + 2x_2 \\ \text{p.o.} \quad & x_1 + x_2 \leq 4 \\ & 2x_1 + x_2 \leq 6 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{2}$$

Ulazni parametri:

```
A1 = [1.0 1.0; 2.0 1.0]
b1 = [4.0, 6.0]
c1 = [3.0, 2.0]
```

Ručno rješavanje - Iteracija po iteracija:
Početna Simplex tabela:

Bazna var	x_1	x_2	s_1	s_2	RHS
s_1	1	1	1	0	4
s_2	2	1	0	1	6
Z	-3	-2	0	0	0

Tablica 1: Početna tabela - Test primjer 1

Ulazna varijabla: x_1 (najnegativniji koeficijent: -3)
Ratio test: $\min\{4/1, 6/2\} = \min\{4, 3\} = 3$
Izlazna varijabla: s_2 (red 2)
Iteracija 1:
Nakon pivot operacije sa pivot elementom $a_{2,1} = 2$:

Bazna var	x_1	x_2	s_1	s_2	RHS
s_1	0	0.5	1	-0.5	1
x_1	1	0.5	0	0.5	3
Z	0	-0.5	0	1.5	9

Tablica 2: Tabela nakon iteracije 1

Ulazna varijabla: x_2 (najnegativniji koeficijent: -0.5)
Ratio test: $\min\{1/0.5, 3/0.5\} = \min\{2, 6\} = 2$
Izlazna varijabla: s_1 (red 1)
Iteracija 2 (finalna):

Bazna var	x_1	x_2	s_1	s_2	RHS
x_2	0	1	2	-1	2
x_1	1	0	-1	1	2
Z	0	0	1	1	10

Tablica 3: Finalna optimalna tabela

Svi koeficijenti u Z redu su nenegativni $\Rightarrow$ optimalno rješenje!
Rješenje:

- $x_1 = 2$, $x_2 = 2$
- $Z = 10$
- Status: OPTIMAL

Poređenje sa izlazom programa:
Izlaz programa se u potpunosti podudara sa ručnim proračunom. Program je prošao kroz 2 iteracije i došao do istog rješenja: $x = [2.0, 2.0]$, $Z = 10.0$.

4.2 Test primjer 2: Problem sa degeneracijom

Formulacija problema:

$$\begin{aligned} \max \quad & Z = x_1 + x_2 \\ \text{p.o.} \quad & x_1 + x_2 \leq 2 \\ & 2x_1 + x_2 \leq 4 \\ & x_1 \leq 2 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{3}$$

Ulazni parametri:

```
A2 = [1.0 1.0; 2.0 1.0; 1.0 0.0]
b2 = [2.0, 4.0, 2.0]
c2 = [1.0, 1.0]
```

Rješenje:

- $x_1 = 2$, $x_2 = 0$
- $Z = 2$
- Status: **DEGENERATE**

Objašnjenje degeneracije:
Degeneracija se javlja kada je neka bazna varijabla jednaka nuli. U ovom primjeru, optimalno rješenje ima $x_2 = 0$ u bazi, što predstavlja degenerisano bazno rješenje. Ovo se može desiti kada se ograničenja sijeku u istoj tački ili kada postoji redundantno ograničenje. Program je uspješno detektovao degeneraciju provjeravajući da li je neka bazna varijabla manja od 10^{-10}.

4.3 Test primjer 3: Beskonačno rješenje (neograničen problem)

Formulacija problema:

$$\begin{aligned} \max \quad & Z = x_1 + x_2 \\ \text{p.o.} \quad & -x_1 + x_2 \leq 1 \\ & x_1 - x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{4}$$

Ulazni parametri:

```
A3 = [-1.0 1.0; 1.0 -1.0]
b3 = [1.0, 1.0]
c3 = [1.0, 1.0]
```

Rješenje:

- $x = [0, 0]$ (nije definisano)
- $Z = \infty$
- Status: **UNBOUNDED**

Objašnjenje beskonačnog rješenja:

Problem je neograničen jer funkcija cilja može rasti u beskonačnost bez narušavanja ograničenja. Kod detektuje ovu situaciju kada su svi koeficijenti u pivot koloni (koloni ulazne varijable) nenegativni ili jednaki nuli. To znači da povećanje ulazne varijable neće narušiti nijedno ograničenje, pa funkcija cilja može rasti neograničeno.

U ovom primjeru, ograničenja ne ograničavaju dovoljno prostor rješenja u smjeru kojim funkcija cilja raste, što rezultira beskonačnim rješenjem.

5 Diskusija o korištenju AI alata

5.1 Pozitivne strane

1. **Brza početna implementacija**: AI alat (Claude) je generisao funkcionalan kod u vrlo kratkom vremenu, što je omogućilo fokus na testiranje i razumijevanje algoritma umjesto na sintaksu.

2. **Dobro strukturiran kod**: Kod je bio čitljiv, dobro komentarisan i organizovan u logične cjeline (inicijalizacija, pivot operacije, provjera optimalnosti).

3. **Podrška za edge cases**: AI je automatski uključio provjere za beskonačna rješenja i degeneraciju, što bi možda bilo zaboravljeno pri ručnoj implementaciji.

4. **Pomoć u razumijevanju**: Objašnjenja u komentarima koda su pomogla u boljem razumijevanju svakog koraka Simplex metode.

5. **Funkcija za ispis tabele**: Generisana je i pomocna funkcija `print_tableau` koja omogućava pregledan prikaz svake iteracije, što olakšava praćenje algoritma.

5.2 Negativne strane

1. **Potreba za verifikacijom**: Kod je morao biti pažljivo testiran i upoređen sa ručnim proračunima. AI ne garantuje tačnost implementacije.

2. **Razumijevanje koda**: Postoji opasnost da student kopira kod bez potpunog razumijevanja svake linije. Potrebno je dodatno vrijeme za analizu.

3. **Terminološke razlike**: AI ponekad koristi različitu terminologiju od one koja se koristi na predavanjima (npr. "reduced costs" umjesto "koeficijenti funkcije cilja").

4. **Potencijalne greške u edge cases**: Mora se voditi računa o numeričkoj stabilnosti i graničnim slučajevima koje AI možda nije adekvatno pokrio.
5. **Ograničeno prilagođavanje**: Kod je morao biti dodatno modifikovan da odgovara tačno specifikacijama zadatka i pseudokodu sa vježbe.

5.3 Da li je AI pomogao u razumijevanju algoritma?

Da, ali sa rezervom. AI alat je omogućio brzu implementaciju koja se mogla testirati i analizirati, što je olakšalo razumijevanje kako Simplex metoda radi u praksi. Međutim, pravo razumijevanje je došlo tek nakon ručnog rješavanja primjera i poređenja sa izlazom programa iteracija po iteracija.

AI alat je najbolje koristiti kao pomoćno sredstvo koje ubrzava proces, ali ne kao zamjenu za teorijsko razumijevanje i ručno vježbanje algoritma. Kombinacija AI alata za implementaciju i vlastitog ručnog rada za verifikaciju pokazala se kao najefikasniji pristup.

6 Zaključak

Laboratorijska vježba je uspješno demonstrirala implementaciju Simplex metode u Juliji uz pomoć AI alata. Sva tri test primjera su dala očekivane rezultate:

- Primjer 1: Optimalno rješenje ($x_1 = 2, x_2 = 2, Z = 10$)
- Primjer 2: Degenerisano rješenje ($x_1 = 2, x_2 = 0, Z = 2$)
- Primjer 3: Beskonačno rješenje (status: UNBOUNDED)

Poređenje ručnih proračuna sa izlazom programa je pokazalo potpunu saglasnost, što potvrđuje ispravnost implementacije. Korištenje AI alata je olakšalo proces, ali je bilo neophodno kritičko razumijevanje i verifikacija rezultata.