Sistemi linearnih jednačina u programskom jeziku Julia

Danilo Kaćanski

Decembar, 2025.

Cilj ovog materijala je upoznavanje sa osnovnim operacijama nad matricama, formiranjem i rešavanjem linearnih sistema jednačina, kao i numeričkim metodama za približno rešavanje problema linearnog modelovanja. Pored teorijske osnove, poseban akcenat stavljen je na primenu u programskom jeziku *Julia*, uz jasnu vezu između matematičkih izraza i numeričke implementacije.

1. Vektori i matrice

U mnogim oblastima modelovanja i simulacije sistema polazna tačka jeste formiranje matrica parametara i vektora stanja. Matrica se u Juliji definiše unosom elemenata po vrstama, a standardni paket `LinearAlgebra` omogućava bogat skup alata za rad sa njima.

```
using LinearAlgebra

A = [1 2 3;
     4 5 6;
     7 8 10]

b1 = [1, 2, 3]          # Vector (3-element)  → posmatramo ga kao kolonu
b2 = [1 2 3]            # Matrix 1x3          → vektor-red
b3 = [1; 2; 3]          # Vector (3-element)  → isto kao b1
```

U prikazanom primeru matrica A je dimenzija 3×3. Vektori `b1` i `b3` su tipa `Vector{Int64}`, što znači da su jednodimenzioni nizovi (3-elementni vektori). U okviru linearne algebre, `Julia` takve vektore uvek tumači kao **vektore–kolone**. Zato je izraz $A * b1$ ispravan matrično–vektorski proizvod.
S druge strane, izraz `[1 2 3]` pravi pravu matricu dimenzija 1×3, tj. vektor–red:

- `[1, 2, 3]` → `Vector` (3 elementa) – posmatra se kao kolona,
- `[1; 2; 3]` → takođe `Vector` (3 elementa) – ista struktura kao gore,
- `[1 2 3]` → `Matrix` dimenzija 1×3 (vektor–red).

Važna napomena je da korišćenje zareza ili tačka–zarez unutar `[...]` ne menja činjenicu da `[1, 2, 3]` i `[1; 2; 3]` prave 1D vektor. Tek upotrebom razmaka

(za kolone) i prelaska u novi red (za vrste) dobijamo eksplicitnu matricu, kao u primeru `[1 2 3]` (matrica 1×3).

2. Transponovanje

U Juliji postoje dva operatora koja su povezana sa transponovanjem matrica:

$$A^T = \text{transpose}(A), \qquad A^* = A'$$

Ovde važi:

- A^T — **obično (klasično) transponovanje**: menjaju se mesta redovima i kolonama, elementi ostaju isti;
- A^* — **konjugovano transponovanje (Hermitovo)**: matrica se prvo transponuje, a zatim se nad svakim elementom uzima kompleksna konjugacija:

$$A^* = \overline{A}^T.$$

Za **realne matrice** nema imaginarnih delova, pa je kompleksna konjugacija ista kao i sama matrica. Zato tada važi

$$A^* = A^T, \qquad A \in \mathbb{R}^{m \times n},$$

ali to **ne znači** da je $A^T = A$. Jednakost

$$A^T = A$$

je tačna samo ako je matrica A **simetrična**.

```
A = [1  2;
     3  4]

B = [1+2im   3-im;
      2im     4-3im]

AT  = transpose(A)   # obi no transponovanje
AH  = A'             # konjugovano transponovanje (isto kao AT jer je A realna)

BT  = transpose(B)   # samo zamena redova i kolona
BH  = B'             # transponovanje + kompleksna konjugacija
```

Kada se koristi obično, a kada konjugovano transponovanje?

- **Obično transponovanje** A^T koristi se kada želimo samo da zamenimo redove i kolone matrice. Koristi se u:
  - geometrijskim transformacijama,
  - formiranju Jacobija ili Hessian matrica,
  - formulisanju sistema linearnih jednačina,

  - radu sa realnim podacima (mehanički sistemi, regulacija, robotika).

- **Konjugovano transponovanje (Hermitovo)** A^* koristi se kada radimo sa **kompleksnim matricama**. Ovo je standardni operator u:
  - obradi signala (Fourier transformacije, spektralne analize),
  - teoriji sistema sa kompleksnim svojstvenim vektorima,
  - kvantnoj mehanici,
  - formiranju Hermitovih matrica ($A^* = A$),
  - računanju unutrašnjeg proizvoda:

    $$\langle x, y \rangle = x^* y,$$

    koji mora biti realan i nenegativan.

Zašto je potrebna konjugacija?
Kod kompleksnih brojeva, samo transponovanje nije dovoljno da bi se sačuvala očekivana svojstva unutrašnjeg proizvoda i energije signala. Konjugacija je potrebna da bi:

- unutrašnji proizvod bio realan i pozitivan,
- norme i energije signala bile pravilno definisane,
- svojstvene vrednosti Hermitovih matrica bile realne,
- Fourier koeficijenti davali fizički ispravna rešenja.

Bez kompleksne konjugacije izgubila bi se simetrija mnogih fundamentalnih svojstava u linearnim i dinamičkim sistemima.

3. Rešavanje linearnih sistema

Linearni sistemi jednačina predstavljaju jedan od najčešćih problema u inženjerskim primenama. Opšti oblik sistema je:

$$Ax = b,$$

gde je $A \in \mathbb{R}^{m \times n}$ matrica koeficijenata, b vektor poznatih veličina, a x nepoznati vektor koji rešavamo.

3.1 Rešavanje pomoću operatora \

U Juliji se linearni sistemi rešavaju operatorom \, koji interno koristi **faktorizaciju matrice**. Faktorizacija znači da se matrica A rastavlja na proizvod jednostavnijih matrica, čime se sistem $Ax = b$ rešava brže i numerički stabilnije nego korišćenjem inverza.
Najčešće korišćene faktorizacije su:

- **LU faktorizacija** — za opšte kvadratne matrice; $A = LU$.
- **QR faktorizacija** — za nekvadratne matrice i metode najmanjih kvadrata; $A = QR$.
- **Cholesky faktorizacija** — za simetrične pozitivno definisane matrice; $A = LL^T$.

Julia automatski bira odgovarajuću faktorizaciju, tako da korisnik jednostavno poziva `A \ b`, dok se optimalna numerička metoda bira u pozadini.
Primer:

```
A = [3  2  -1;
     2 -2   4;
    -1  0.5 -1]

b = [1, -2, 0]

x = A \ b   # resavanje sistema
```

3.2 Zašto ne koristiti `inv(A)` za rešavanje sistema?

Inverzija matrice se ne koristi iz dva razloga:

1. **Numerička nestabilnost** — računanje inverza pojačava greške zaokruživanja, naročito kada je matrica slabo uslovljena.
2. **Skuplja računarski** — izračunavanje `inv(A)` zahteva više operacija nego direktno rešavanje sistema.

Loš pristup:

```
x_wrong = inv(A) * b   # matematicki ispravno, numericki lose
```

U praksi se `inv(A)` koristi gotovo isključivo za teorijske izvedbe, ne za numeričko rešavanje.

3.3 Provera tačnosti rešenja

Nakon izračunavanja rešenja, korisno je proveriti da li važi:

$$Ax \approx b.$$

U Juliji:

```
A * x ≈ b    # proverava jednakost uz toleranciju
```

3.4 Tipovi linearnih sistema: tačno određen, predodređen i neodređen

Linearan sistem $Ax = b$ može imati različit broj jednačina (m) i nepoznatih (n). Od odnosa m i n zavisi kako se sistem rešava i da li ima jedno, više ili nijedno rešenje.

1) Tačno određen sistem ($m = n$)

Jednak broj jednačina i nepoznatih.

- Ako je $\det(A) \neq 0$, postoji **jedinstveno rešenje**.
- Julia koristi LU faktorizaciju pri rešavanju.

Primer u Juliji:

```
A = [3 1;
     2 4]

b = [7, 10]
x = A \ b       # jedinstveno resenje
```

2) Predodređen (overdetermined) sistem ($m > n$)

Više jednačina nego nepoznatih — tipično nema tačno rešenje.

- Traži se **rešenje u smislu najmanjih kvadrata**:

$$x = \arg\min_x \|Ax - b\|.$$

- Julia automatski koristi QR faktorizaciju pri pozivu `A \ b`.

Primer u Juliji:

```
A = [1 1;
     1 2;
     1 3]     # 3 jednacine, 2 nepoznate (m > n)

b = [1, 2, 2.2]

x = A \ b     # LS resenje
```

3) Neodređen (underdetermined) sistem ($m < n$)

Manje jednačina nego nepoznatih — sistem ima beskonačno mnogo rešenja.

- Julia bira **rešenje minimalne norme**:

$$x = \arg\min \|x\| \quad \text{uz uslov } Ax = b.$$

- Interno se koriste QR ili SVD faktorizacije.

Primer u Juliji:

```
A = [1 2 3]          # 1 jednacina, 3 nepoznate (m < n)
b = [10]

x = A \ b            # minimal-norm resenje
```

Dakle, iako korisnik uvek poziva izraz `A \ b`, način rešavanja zavisi od odnosa dimenzija matrice A. Julia automatski bira numerički najstabilniju i najefikasniju metodu.

4. Determinanta i rang matrice

4.1 Determinanta

Determinanta matrice $A \in \mathbb{R}^{n\times n}$ označava se sa $\det(A)$ i predstavlja meru *zapremine* koju formiraju kolone matrice.
Ključne činjenice:

- Ako je $\det(A) \neq 0$, matrica je **nesingularna** (invertibilna).
- Ako je $\det(A) = 0$, matrica je **singularna** i ne postoji jedinstveno rešenje sistema $Ax = b$.
- Determinanta nula znači da su kolone ili vrste linearno zavisne.

Primer u Juliji:

```
A1 = [1 2;
      3 4]

A2 = [1 2;
      2 4]   # drugi red je 2x prvi

detA1 = det(A1)   # = -2, matrica je invertibilna
detA2 = det(A2)   # = 0, matrica je singularna
```

Tumačenje:

- A_1 ima nezavisne kolone $\rightarrow$ sistem $A_1x = b$ ima jedinstveno rešenje.
- A_2 ima zavisne kolone $\rightarrow$ beskonačno mnogo ili nijedno rešenje.

4.2 Rang matrice

Rang matrice predstavlja broj linearno nezavisnih vrsta ili kolona.

$$\operatorname{rank}(A) = r$$

Osnovne činjenice:

- Ako je $\operatorname{rank}(A) = n$ (za kvadratnu $n \times n$ matricu), onda je matrica invertibilna.

- Ako je $\text{rank}(A) < n$, matrica je singularna — kolone su zavisne.
- Rang određuje broj nezavisnih jednačina u sistemu $Ax = b$.

Primeri:

```
B1 = [1 2 3;
      4 5 6;
      7 8 10]

B2 = [1 2 3;
      2 4 6;
      3 6 9]

rankB1 = rank(B1)   # = 3, pune ranga
rankB2 = rank(B2)   # = 1, sve vrste zavisne
```

Tumačenje:

- B_1 ima rang 3 → linearno nezavisne vrste → sistem ima jedinstveno LS rešenje.
- B_2 ima rang 1 → samo jedna nezavisna jednačina, ostale zavisne.

5. Metoda najmanjih kvadrata

U praksi često imamo **više jednačina nego nepoznatih**, tj. sistem oblika:

$$Ax \approx b, \qquad A \in \mathbb{R}^{m\times n},\ m > n.$$

Takav sistem je **nadodređen**: većina sistema $Ax = b$ *nema tačno rešenje*. Zbog toga se traži vektor x koji najbolje *prilazi* vektoru b.

5.1 Ideja metode najmanjih kvadrata

Umesto tačnog zadovoljenja $Ax = b$, traži se x koje minimizuje grešku:

$$x = \arg\min_x \|Ax - b\|_2.$$

Geometrijski: Ax je projekcija vektora b na prostor kolona matrice A.
Rešenje je dato formulom:

$$x = (A^T A)^{-1} A^T b,$$

ali se u praksi **nikada ne računa direktno**, već se koristi QR faktorizacija, koju Julia automatski primenjuje pri pozivu `A \ b`.

5.2 Primer: Aproksimacija pravom linijom

Neka su poznate merene vrednosti funkcije $y(t)$:

$$y \approx at + b.$$

To možemo zapisati kao linearni sistem:

$$A = \begin{bmatrix} t_1 & 1 \\ t_2 & 1 \\ t_3 & 1 \\ \vdots & \vdots \\ t_m & 1 \end{bmatrix}, \qquad x = \begin{bmatrix} a \\ b \end{bmatrix}.$$

Primer 1. Aproksimacija prave linije realnim podacima.

```
using LinearAlgebra, Plots

t = [0, 1, 2, 3]
y = [1.0, 2.1, 3.0, 4.1]   # realna merenja

A = [t  ones(length(t))]   # matrica A = [t  1]

x = A \ y    # LS resenje → koeficijenti [a, b]
a, b = x
```

Model funkcije:

$$\hat{y}(t) = at + b.$$

5.3 Plotovanje rezultata

Poredi se:

- originalni podaci (tačke),
- aproksimaciona prava linija,
- reziduali (greske) $r = Ax - b$.

```
t_fine = 0:0.01:3   # glatka linija za crtanje
y_fit  = a .* t_fine .+ b

# Reziduali
r = A*x .- y

p1 = scatter(t, y, label="originalna merenja", legend=:topleft)
plot!(p1, t_fine, y_fit, lw=3, label="LS aproksimacija")

p2 = bar(r, label="reziduali (Ax - b)",
         xlabel="i", ylabel="greska")

plot(p1, p2, layout=(2,1), size=(650,600))
```

Rezultat prikazuje:

- koliko se model udaljava od stvarnih merenja,
- da LS minimizuje sumu kvadrata odstupanja,

- kako aproksimaciona funkcija izgleda u odnosu na podatke.

5.4 Intuicija zašto se minimizuje suma kvadrata

Za svaku tačku merenja imamo grešku:

$$r_i = at_i + b - y_i.$$

Želimo da sistem bude što tačniji, ali ne možemo tačno zadovoljiti sve jednačine. Zato minimizujemo:

$$\|Ax - b\|_2^2 = \sum_{i=1}^{m} r_i^2.$$

Kvadrati grešaka se koriste jer:

- kažnjavaju velika odstupanja više nego mala,
- daju jedinstveno i glatko rešenje,
- omogućavaju zatvorenu formulu sa $A^T A$.

5.5 Drugi primer: Polinom drugog stepena

Model:

$$y(t) \approx c_1 t^2 + c_2 t + c_3.$$

```
t = 0:0.5:5
y = 1 .+ 0.5 .* t .- 0.1 .* t.^2 .+ 0.2*randn(length(t))   # simulirani podaci

A = [t.^2  t  ones(length(t))]

coef = A \ y   # [c1, c2, c3]
```

Julia automatski radi LS i pronalazi najbolji polinom drugog stepena.

6. Inverzija matrica

Iako se u numeričkoj praksi inverzija **retko koristi za rešavanje sistema** (zbog nestabilnosti i veće računarske složenosti), ona je i dalje važan matematički koncept.

6.1 Definicija

Za kvadratnu matricu A koja je regularna (nesingularna, $\det(A) \neq 0$), inverzna matrica A^{-1} definiše se uslovom:

$$A^{-1}A = I, \qquad AA^{-1} = I,$$

gde je I matrica identiteta.

6.2 Kada inverzija postoji?

- Inverzija postoji samo ako je A **kvadratna** i **punog ranga**.
- Ako je $\det(A) = 0$, matrica je singularna i A^{-1} **ne postoji**.
- Ako je A loše uslovljena, inverzija postoji, ali je numerički nepouzdana.

6.3 Računanje inverza u Juliji

```
A = [1 2 3;
     0 1 4;
     2 0 1]

A_inv = inv(A)
```

Može se proveriti da važi približna jednakost:

```
A * A_inv ≈ I
```

6.4 Zašto se inverzija ne koristi za rešavanje sistema?

- numerički **nestabilnija**: greške zaokruživanja se pojačavaju,
- računarski **skuplja**: zahteva više operacija nego faktorizacija,
- nepotrebna: Julia već optimalno rešava $Ax = b$ bez inverza.

Zato se izraz

$$x = A^{-1}b$$

skoro nikada ne koristi u numeričkim primenama, već se rešavanje vrši pomoću:

$$x = A\backslash b.$$

7. Zadaci

Zadatak 1. Za datu matricu

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 10 \end{bmatrix},$$

izračunati:

a) transponovanu matricu A^T i Hermitovu matricu A^*,

b) rang matrice,

c) determinantu,

d) svojstvene vrednosti matrice.

Zadatak 2. Neka je sistem $Ax = b$ definisan matricom

$$A = \begin{bmatrix} 2 & -1 \\ 5 & 3 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 14 \end{bmatrix}.$$

a) rešiti sistem korišćenjem operatora \,

b) proveriti da li važi $Ax \approx b$ i izračunati vektor reziduala $r = Ax - b$,

c) izračunati normu reziduala $\|r\|_2$.

Zadatak 3. Dat je sistem definisan matricom

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 2 \\ 2.1 \end{bmatrix}.$$

Odrediti:

a) da li je sistem tačno određen, predodređen ili neodređen,

b) kakvo se rešenje očekuje (jedinstveno, LS rešenje ili minimal-norm rešenje),

c) rešiti sistem u Juliji i komentar dati na osnovu dobijenog rešenja.

Zadatak 4. Neka je matrica

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 1 & 1 & 1 \end{bmatrix}.$$

a) odrediti rang matrice,

b) proveriti da li je matrica singularna,

c) objasniti kakvu implikaciju to ima na rešavanje sistema $Ax = b$.

Zadatak 5. Metodom najmanjih kvadrata aproksimirati linearnu funkciju:

$$y(t) \approx at + b$$

za 10 tačaka generisanih izrazom:

$$y(t) = 3t + 5 + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, 0.2).$$

Zatim:

a) nacrtati originalne tačke i dobijenu LS pravu,

b) izračunati i prikazati reziduale,

c) izračunati normu greške $\|Ax - b\|_2$.

Zadatak 6. Razmotriti neodređeni sistem:

$$A = [1 \;\; 2 \;\; 3], \qquad b = [10].$$

a) rešiti sistem izrazom `A \ b`,

b) izračunati normu rešenja $\|x\|_2$,

c) objasniti zašto je Julia vratila baš minimal-norm rešenje.

Literatura

- Gilbert Strang: *Linear Algebra and Its Applications*
- Trefethen & Bau: *Numerical Linear Algebra*
- Dokumentacija programskog jezika Julia — https://julialang.org