Numeričko rješavanje linearnih sustava

Mikec, Karla

Master's thesis / Diplomski rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:967905

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-16**

Repository / Repozitorij:

Repository of the Faculty of Science - University of Zagreb

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO–MATEMATIČKI FAKULTET
MATEMATIČKI ODSJEK

Karla Mikec

NUMERIČKO RJEŠAVANJE LINEARNIH SUSTAVA

Diplomski rad

Voditelj rada:
doc. dr. sc. Marko Erceg

Zagreb, rujan, 2018.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Ovaj rad posvećujem svojim roditeljima bez kojih moje postignuće ne bi bilo moguće. Ujedno zahvaljujem njima, kao i cijeloj obitelji, na pruženoj potpori i razumijevanju tijekom cijelog studija.

Posebna zahvala mentoru doc. dr. sc. Marku Ercegu što ste uvijek našli vremena za moja pitanja te mi svojim savjetima pomogli u izradi rada.

Sadržaj

Sadržaj **iv**

Uvod **1**

1 Uvod u linearne sustave **2**
1.1 Svojstva linearnih sustava 2
1.2 Normirani prostori 5

2 Numeričko rješavanje linearnih sustava: direktne metode **7**
2.1 Gaussova metoda eliminacije 7
2.1.1 Opis metode 7
2.1.2 Algoritam 9
2.1.3 Broj operacija 10
2.2 LR faktorizacija 13
2.2.1 Opis metode 13
2.2.2 Algoritam 17
2.2.3 Računanje inverza 19

3 Numeričko rješavanje linearnih sustava: iterativne metode **21**
3.1 Općenito o iterativnim metodama 21
3.2 Jacobijeva metoda 24
3.2.1 Opis metode 24
3.2.2 Algoritam 25
3.3 Gauss-Seidelova metoda 26
3.3.1 Opis metode 26
3.3.2 Algoritam 27
3.4 Uvjeti konvergencije 28
3.5 Osnovni primjeri 30
3.6 Numeričko rješavanje diferencijalne jednadžbe 33

Bibliografija **42**

Uvod

Rješavanje sustava linearnih jednadžbi jedno je od osnovnih matematičkih znanja koje se proteže od osnovnoškolskog pa sve do visokoškolskog matematičkog obrazovanja. Za njihovo rješavanje postoji nekoliko metoda koje nisu teške za provesti kada su dani manji sustavi. No, s povećanjem broja jednadžbi i broja nepoznanica, postupak postaje kompliciraniji i puno sporiji. Upravo zbog toga je problem sustava linearnih jednadžbi zanimljiv numeričkoj matematici koja nastoji dati što preciznije i brže metode njihovog rješavanja.

U prvom poglavlju ovog rada su dana osnovna svojstva linearnih algebarskih sustava, te su uvedeni pojmovi koji će se koristiti u radu. U drugom poglavlju su promatrane direktne metode za rješavanje linearnih sustava, i to Gaussova metoda eliminacija i LR (ili LU) faktorizacija. U zadnjem poglavlju su opisane iterativne metode od kojih su izdvojene Jacobijeva i Gauss-Seidelova metoda, te je prikazano na koji način se diferencijalne jednadžbe svode na rješavanje linearnih sustava.

Svaka metoda je ukratko opisana, zapisana pomoću algoritma u programskom jeziku Python 3.6 te potkrijepljena primjerom.

Poglavlje 1

Uvod u linearne sustave

U ovom poglavlju dajemo uvodne definicije i teoreme iz [1], [2] i [4].

1.1 Svojstva linearnih sustava

Opći sustav linearnih jednadžbi nad poljem $\mathbb{F}$ je sustav od m linearnih jednadžbi s n nepoznanica, oblika

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{m1}x_1 & + & a_{m2} & + & \cdots & + & a_{mn}x_n & = & b_m\,, \end{array} \tag{1.1}$$

pri čemu su $m, n \in \mathbb{N}$. Riješiti sustav linearnih jednadžbi znači odrediti skup svih rješenja sustava pri čemu je jedno rješenje svaka uređena n-torka $(\gamma_1, \gamma_2, \ldots, \gamma_n) \in \mathbb{F}$ koja je rješenje svake pojedine jednadžbe sustava.

Sustav možemo zapisati matrično definiramo li *matricu sustava*

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix},$$

čije elemente nazivamo *koeficijenti sustava*, zatim *matricu nepoznanica*

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},$$

te *matricu slobodnih članova*

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$$

Koristeći navedene oznake, sustav (1.1) možemo ekvivalentno zapisati u obliku

$$Ax = b\,. \tag{1.2}$$

U ovom radu baviti ćemo se sustavima u kojima je matrica A *kvadratna matrica*, što znači da je $m = n$. Također, matrice će biti *regularne*. Definirajmo takve matrice i sustave.

Definicija 1.1.1. *Kažemo da je kvadratna matrica A reda n regularna ako postoji kvadratna matrica B reda n takva da vrijedi*

$$AB = BA = I\,,$$

pri čemu je I jedinična matrica reda n. Matricu B tada zovemo inverzna matrica i označavamo s A^{-1}.

Definicija 1.1.2. *Kažemo da je sustav $Ax = b$ Cramerov ako je A kvadratna regularna matrica.*

Teorem 1.1.3. *[1, Propozicija 4.2.8] Cramerov sustav $Ax = b$ je rješiv, a rješenje mu je jedinstveno i dano formulom*

$$C = A^{-1}B\,.$$

Dakle, sve metode dane u ovom radu biti će za rješavanje Cramerovih sustava linearnih jednadžbi koji uvijek imaju jedinstveno rješenje. Kao što vidimo iz Teorema 1.1.3, takve sustave možemo rješavati izračunamo li inverznu matricu A^{-1} matrice A kojom pomnožimo relaciju (1.2) slijeva te dobivamo rješenje

$$x = A^{-1}b\,.$$

No, računanje inverza je kompliciraniji problem od rješavanja linearnog sustava jednadžbi pa se u praksi najčešće koristi metoda Gaussovih eliminacija kojom se sustav svodi na trokutastu formu. Zapišimo definicije i tvrdnje koje će nam biti potrebne za postupak Gaussovih eliminacija koji je opisan u sljedećem poglavlju.

Definicija 1.1.4. *Dva sustava linearnih jednadžbi nad poljem $\mathbb{F}$ su ekvivalentna ako imaju isti broj nepoznanica i isti skup rješenja.*

Definicija 1.1.5. *Elementarne transformacije sustava linearnih jednadžbi su:*
(i) zamjena poretka dviju jednadžbi,
(ii) množenje neke jednadžbe skalarom $\lambda \neq 0$,
(iii) pribrajanje neke jednadžbe pomnožene skalarom λ nekoj drugoj jednadžbi sustava.

Teorem 1.1.6. *Primjenom konačnog broja elementarnih transformacija na dani sustav linearnih jednadžbi dobiva se ekvivalentan sustav.*

Dokaz. Pokažimo da su sustavi $Ax = b$ i $A_1x = b_1$ ekvivalentni, pri čemu drugi sustav nastaje iz prvog primjenom jedne od elementarnih transformacija. Da bismo to pokazali, dovoljno je pokazati da je neko proizvoljno rješenje sustava

$$Ax = b\,,$$

također rješenje sustava

$$A_1x = b_1\,.$$

Ako smo za dobivanje sustava $A_1x = b_1$ primijenili transformaciju (i) ili (ii), tada je tvrdnja teorema trivijalna i rješenje dobivenog sustava je jednako rješenju početnog sustava. Pretpostavimo da smo primijenili transformaciju (iii), odnosno pomnožili i-tu jednadžbu početnog sustava s λ i dodali k-toj jednadžbi. Također neka je $(\gamma_1, \gamma_2, \ldots, \gamma_n)$ proizvoljno rješenje početnog sustava. Uočimo da se početni i dobiveni sustav razlikuju samo u k-toj jednadžbi pa trebamo provjeriti da rješenje $(\gamma_1, \gamma_2, \ldots, \gamma_n)$ zadovoljava upravo k-tu jednadžbu dobivenog sustava $A_1x = b_1$. Imamo

$$\sum_{j=1}^{n} (\lambda a_{ij} + a_{kj})\,\lambda_j = \lambda \sum_{j=1}^{n} a_{ij}\lambda_j + \sum_{j=1}^{n} a_{kj}\lambda_j = \lambda b_i + b_k\,,$$

čime smo pokazali da je proizvoljno rješenje početnog sustava također rješenje dobivenog sustava, odnosno ti sustavi su ekvivalentni. □

1.2 Normirani prostori

Numeričkim rješavanjem sustava linearnih jednadžbi dolazi se do dovoljno točne aproksimacije rješenja, odnosno javljaju se greške u rezultatu koje ocjenjujemo koristeći norme. Stoga definirajmo pojam norme i normiranog prostora te dajmo primjer nekih normi.

Definicija 1.2.1. *Norma na vektorskom prostoru X nad poljem $\mathbb{F}$ je preslikavanje $\|\cdot\| : X \to \mathbb{R}$ koje ima sljedeća svojstva:*

(i) $\|x\| \geqslant 0$, $\forall x \in X$, *a jednakost vrijedi ako i samo ako je* $x = 0$;

(ii) $\|\alpha x\| = |\alpha|\|x\|$, $\forall \alpha \in \mathbb{F}$, $\forall x \in X$;

(iii) $\|x + y\| \leqslant \|x\| + \|y\|$, $\forall x, y \in X$.

Uređen par $(X, \|\cdot\|)$ *naziva se normiran prostor.*

Za vektor $x \in \mathbb{R}^n$, s komponentama x_i, $i = 1, \ldots, n$, najčešće se koriste l_1 norma

$$\|x\|_1 = \sum_{i=1}^{n} |x| \,,$$

zatim l_2 norma ili euklidska norma

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x|^2} \,,$$

te l_∞ norma

$$\|x\|_\infty = \max_{i=1,\ldots,n} |x| \,.$$

Promatramo li vektorski prostor kvadratnih matrica $M_n(\mathbb{R})$, za matricu $A \in M_n(\mathbb{R})$, na isti način možemo definirati l_1 normu

$$\|A\|_S = \sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}| \,,$$

l_2 ili Frobeniusovu normu

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

te l_∞ normu

$$\|A\|_M = \max_{\substack{i=1,\dots,n \\ j=1,\dots,n}} |a_{ij}| .$$

Naveli smo neke primjere matričnih normi direktno dobivenih iz vektorskih. Općenito, *matričnu normu* definiramo kao funkciju koja zadovoljava sva svojstva iz Definicije 1.2.1, ali uz njih mora zadovoljavati i svojstvo

$$\|AB\| \leqslant \|A\|\|B\|$$

koje zovemo svojstvo *konzistentnosti*. Matrične norme inducirane vektorskom normom $\|\cdot\|$ na $\mathbb{R}^n$ dane su općom definicijom

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|} , \quad x \in \mathbb{R}^n ,$$

a uvrštavanjem l_1, l_2 i l_3 norme dobivamo matričnu 1-normu

$$\|A\|_1 = \max_{j=1,\dots,n} \sum_{i=1}^{m} |a_{ij}| ,$$

matričnu 2-normu ili spektralnu normu

$$\|A\|_2 = \sqrt{\rho(A^* A))}$$

kod koje je ρ oznaka za spektralni radijus kvadratne matrice te matričnu ∞-normu

$$\|A\|_\infty = \max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}| .$$

Definicija 1.2.2. *Kažemo da je normiran prostor potpun ako svaki Cauchyjev niz u njemu konvergira. Potpun normiran prostor zovemo još i Banachov prostor.*

Definicija 1.2.3. *Kažemo da su norme $\|\cdot\|_a$ i $\|\cdot\|_b$ definirane na vektorskom prostoru X ekvivalentne ako postoje m, $M > 0$ takvi da vrijedi*

$$m\|x\|_a \leqslant \|x\|_b \leqslant M\|x\|_a , \quad \forall x \in X .$$

Teorem 1.2.4. *[4, Teorem 1.1.17] Sve norme na konačnodimenzionalnom vektorskom prostoru su ekvivalentne.*

Teorem 1.2.5. *[4, Propozicija 1.2.2] Svaki konačnodimenzionalan normiran prostor nad $\mathbb{R}$ je potpun.*

Poglavlje 2

Numeričko rješavanje linearnih sustava: direktne metode

Za numeričko rješavanje linearnih sustava postoji više metoda koje možemo podijeliti u dvije skupine: direktne metode (Gaussove eliminacije, LR faktorizacija) koje daju točno rješenje sustava unutar konačnog broja koraka ako nema pogrešaka u zaokruživanju i iterativne metode (Jacobijeva, Gauss-Seidelova) koje u svakom koraku pokušavaju bolje aproksimirati rješenje sustava te su korisne za rješavanje velikih sustava linearnih jednadžbi.
U ovom poglavlju pratimo literaturu [2], [3] te [8]. Također, za svaku metodu dati ćemo algoritam napisan u programskom jeziku Python 3.6 u kojem će biti rješavani i svi dani primjeri.

2.1 Gaussova metoda eliminacije

2.1.1 Opis metode

Promatramo linearni sustav jednadžbi oblika $Ax = b$ gdje je A regularna kvadratna matrica reda n, $n \geqslant 2$. Prema Teoremu 1.1.3 znamo da ovaj sustav ima jedinstveno rješenje. Ideja metode Gaussovih eliminacija je transformirati sustav $Ax = b$ na gornje trokutastu formu. Proširena matrica sustava je oblika

$$\left[\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{array}\right].$$

Za dobivanje gornjetrokutaste matrice krećemo s prvim stupcem u kojem želimo da svi elementi, osim prvog, budu nula. Pretpostavimo, dakle, da je $a_{11}^{(1)} \neq 0$ i oduzmemo od drugog retka prvi redak matrice pomnožen s $\dfrac{a_{21}^{(1)}}{a_{11}^{(1)}}$. Time na željenom mjestu dobijemo nulu, a na preostalim mjestima u retku dobijemo nove elemente. Ponovimo postupak sa svakim sljedećim retkom, dakle i-ti redak pomnožimo s $\dfrac{a_{i1}^{(1)}}{a_{11}^{(1)}}$, $i = 2, \ldots, n$, i od njega oduzmemo prvi redak. Dobije se ekvivalentna matrica

$$\left[\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{array}\right]$$

u kojoj je svaki element $a_{ik}^{(2)}$, $i\,, k = 2, \ldots, n$, dobiven pomoću relacije

$$a_{ik}^{(2)} = a_{ik}^{(1)} - m_{i1} a_{1k}^{(1)}\,,$$

a svaki element $b_i^{(2)}$ dobiven je kao

$$b_i^{(2)} = b_i^{(1)} - m_{i1} b_1^{(1)}\,,$$

pri čemu je

$$m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}\,.$$

Sada promatramo drugi stupac matrice i želimo da su svi elementi ispod dijagonale jednaki nuli. Pretpostavimo da je $a_{22}^{(2)} \neq 0$ i na isti način svaki i-ti redak, $i = 3, \ldots, n$, pomnožimo s $\dfrac{a_{i2}^{(2)}}{a_{22}^{(2)}}$ i od njega oduzmemo drugi redak. Nastavimo s postupkom do $n-1$ - og retka i dobivanja gornjetrokutaste matrice oblika

$$\left[\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(n)} & b_n^{(n)} \end{array}\right]\,.$$

Dobiveni sustav je, prema Teoremu 1.1.6, ekvivalentan početnom i lako se rješava jer iz posljednje jednadžbe dobivamo

$$x_n = \frac{b_n^{(n)}}{a_{nn}^{(n)}}\,,$$

što iskoristimo u prethodnoj jednadžbi za dobivanje x_{n-1} i tako supstitucijama unazad dolazimo konačno do x_1. Dakle, svaki prethodni x_i dobije se pomoću relacije

$$x_i = \frac{1}{a_{ii}^{(i)}} \left(b_i^{(i)} - \sum_{j=i+1}^{n} a_{ij}^{(i)} x_j \right) , \quad i = n-1, n-2, \ldots, 1 .$$

Uočimo da smo u svakom koraku ovog postupka pretpostavljali da je element na dijagonali $a_{ii}^{(i)}$, kojim smo dijelili, različit od nule. Taj element zovemo *pivotni element*, a postupak kojim osiguravamo da on nije nula zovemo *pivotiranje*. Mi ćemo koristiti parcijalno pivotiranje, odnosno zamjenu redaka matrice. Dakle, ako je element $a_{11}^{(1)}$ matrice A jednak nuli, možemo zamijeniti prvi redak s nekim od preostalih redaka kojem element $a_{i1}^{(1)}$, $i = 2, \ldots, n$, nije nula. Znamo da takav element postoji jer kad bi svaki od elemenata u prvom stupcu matrice bio nula, tada matrica ne bi bila regularna. Tim tranformacijama i dalje imamo sustav ekvivalentan početnome. Nastavljamo postupak ako je bilo koji od sljedećih pivotnih elemenata jednak nuli.

Osim osiguravanja ne-nul elemenata, postupkom pivotiranja smanjuju se pogreške zaokruživanja u računalu ako se kao pivotni element u stupcu uzima najveći po apsolutnoj vrijednosti gledajući od elementa na dijagonali i ispod nje. Dakle, ako je u k-tom koraku

$$|a_{rk}^{(k)}| = \max_{k \leqslant i \leqslant n} |a_{ik}^{(k)}|,$$

onda mijenjamo r-ti i k-ti redak te je $a_{rk}^{(k)}$ pivotni element kojim dalje nastavljamo Gaussove eliminacije.

2.1.2 Algoritam

Prema navedenom postupku Gaussovih eliminacija s parcijalnim pivotiranjem zapišimo algoritam u Pythonu za rješavanje sustava $Ax = b$ gdje su nam A i b zadani ulazni podaci, a konačno rješenje x zapisano je u b.

Algoritam 2.1.1.

```
1  def gauss(A,b):
2      n=len(b)
3      for k in range (0,n-1):
4          max_elt=0
5          ind_max=k
6          for i in range (k, n):          #tražimo maksimalni element
7              if abs(A[i][k])>max_elt:
```

```
8                   max_elt=abs(A[i][k])
9                   ind_max=i
10          if max_elt>0:
11              if ind_max>k:                       #pivotiranje
12                  for j in range (k, n):          #mijenjamo k-ti i
13                      temp=A[ind_max][j]          ind_max-ti redak
14                      A[ind_max][j]=A[k][j]
15                      A[k][j]=temp
16                  temp=b[ind_max]
17                  b[ind_max]=b[k]
18                  b[k]=temp
19              for i in range(k+1, n):             #svođenje na trokutastu
20                  m=A[i][k]/A[k][k]               formu
21                  A[i][k]=0
22                  for j in range (k+1, n):
23                      A[i][j]=A[i][j]-m*A[k][j]
24                  b[i]=b[i]-m*b[k]
25          else:
26              print ("Matrica je singularna.")
27              exit()
28      if A[n-1][n-1]!=0:
29          b[n-1]=b[n-1]/A[n-1][n-1]               #supstitucije unazad
30          for i in range (n-2,-1,-1):             #rješenje zapisano u b
31              s=b[i]
32              for j in range (i+1, n):
33                  s=s-A[i][j]*b[j]
34              b[i]=s/A[i][i]
35      else:
36          print ("Matrica je singularna.")
37          exit()
38      return(b)
```

2.1.3 Broj operacija

Zanima nas složenost metode Gaussovih eliminacija koju mjerimo brojem aritmetičkih operacija u navedenom algoritmu. U postupku pivotiranja ne nailazimo na operacije dijeljenja, množenja, zbrajanja i oduzimanja pa brojanje počinjemo od svođenja sustava na trokutastu formu, odnosno linije 19 u Algoritmu 2.1.1. Odredimo najprije broj operacija u proizvoljnom k-tom koraku for petlje u liniji 3. Uočimo da se dijeljenje

(linija 20) izvršava za svaki i iz *for* petlje (linija 19) što je $n-1-(k+1)+1 = n-k-1$ puta. Množenje se javlja u svakom koraku u kojem i oduzimanje (linija 23), odnosno za svaki i javljaju se $n-k-1$ puta zbog *for* petlje (linija 22) te još jednom kod računanja elemenata vektora b (linija 24). Dakle, u tom dijelu imamo $2(n-k-1)(n-k-1+1) = 2(n-k-1)(n-k)$ množenja i oduzimanja što je ukupno

$$\begin{aligned} n-k-1+2(n-k-1)(n-k) &= n-k-1+2n^2-2nk-2nk+2k^2-2n+2k \\ &= 2n^2-4nk+2k^2-n+k-1 \\ &= 2(n-k)^2-(n-k+1) \end{aligned}$$

operacija.

Iz linije 3 uočavamo da k ide od 0 do $n-2$ tako da je ukupan broj aritmetičkih operacija potrebnih za svođenje na trokutasti oblik dan s

$$\sum_{k=0}^{n-2} \left[2(n-k)^2-(n-k+1)\right] .$$

Upotrijebimo li supstituciju $l = n-k$, kada uskladimo granice sume i primijenimo komutativnost zbrajanja dobivamo

$$\sum_{l=2}^{n} \left[2l^2-(l+1)\right] .$$

Dakle, sada imamo

$$\begin{aligned} \sum_{l=2}^{n} \left[2l^2-(l+1)\right] &= \sum_{l=1}^{n-1} \left[2(l+1)^2-(l+2)\right] \\ &= \sum_{l=1}^{n-1} \left(2l^2+3l\right) \\ &= 2\sum_{l=1}^{n-1} l^2 + 3\sum_{l=1}^{n-1} l \\ &= 2\cdot\frac{(n-1)n(2n-1)}{6} + 3\cdot\frac{(n-1)n}{2} \\ &= n(n-1)\left(\frac{2}{3}n+\frac{7}{6}\right) \\ &= \frac{2}{3}n^3+\frac{1}{2}n^2-\frac{7}{6}n \end{aligned}$$

operacija tijekom svođenja sustava na trokutastu formu. Još preostaje izračunati broj aritmetičkih operacija za dobivanje rješenja, odnosno za supstitucije unazad. Na početku imamo jedno dijeljenje (linija 29) i zatim još $n-1$ dijeljenja (linija 34) za svaki $i \in \{0, \ldots, n-1\}$ što je ukupno n dijeljenja. Broj oduzimanja i množenja je jednak (linija 33), a dobivamo ga kao

$$\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$$

što znači da je ukupan broj operacija u ovom dijelu jednak

$$n + 2 \cdot \frac{n(n-1)}{2} = n + n^2 - n = n^2 \,. \tag{2.1}$$

Pribrojimo li to prethodno dobivenom izrazu, imamo da je broj svih operacija Gaussovih eliminacija jednak

$$\frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{7}{6}n + n^2 = \frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{7}{6}n \,.$$

Uočimo da je $\frac{2}{3}n > \frac{3}{2}$ za $n \geqslant 3$, odnosno za velike n je broj operacija približno jednak $\frac{2}{3}n^3$. Dakle, ovaj algoritam ima kubnu složenost pa se za veće sustave linearnih jednadžbi javlja potreba za nekim drugim metodama njihovog rješavanja.

Primjer 2.1.2. *Riješimo sustav $Ax = b$ gdje je*

$$A = \begin{bmatrix} 3 & 2 & 1 & 1 \\ 2 & -1 & 0 & -1 \\ 4 & 3 & 2 & 3 \\ 0 & 5 & 2 & 3 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

koristeći Algoritam 2.1.1.

Unošenje ulaznih podataka i poziv funkcije vršimo na sljedeći način

```
A=[[3,2,1,1],[2,-1,0,-1],[4,3,2,3],[0,5,2,3]]
b=[3,1,1,1]
x=gauss(A,b)
print(x)
```

čime dobivamo sljedeći rezultat

`[1.0, 1.9999999999999996, -2.999999999999999, -1.0]`.

Kako je egzaktno rješenje ovog sustava jednako

$$x = \begin{bmatrix} 1 \\ 2 \\ -3 \\ -1 \end{bmatrix},$$

možemo uočiti da smo našim algoritmom zapravo dobili točno rješenje do na očekivanu malu pogrešku zbog konačne aritmetike računala.

2.2 LR faktorizacija

2.2.1 Opis metode

Postupkom Gaussovih eliminacija možemo matricu A sustava faktorizirati u obliku

$$A = LR\,,$$

gdje je L donjetrokutasta matrica s jedinicama po dijagonali, a R je gornjetrokutasta matrica, odnosno imamo rastav

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & r_{nn} \end{bmatrix}.$$

Sada početni sustav možemo zapisati kao

$$LRx = b\,,$$

a rješenje je jednako

$$x = (LR)^{-1}b = R^{-1}L^{-1}b\,.$$

Označimo li s $y = Rx$, dobijemo dva trokutasta sustava, $Ly = b$ i $Rx = y$ koja vrlo lako rješavamo. U prvom sustavu

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

imamo donjetrokutastu matricu i rješavamo ga supstitucijama unaprijed. Dakle, odmah imamo da je

$$y_1 = b_1 \,,$$

a svaki sljedeći dobijemo uvrštavajući prethodne, odnosno

$$y_i = b_i - \sum_{k=1}^{i-1} l_{ij} y_j \,, \quad i = 2, \ldots, n \,.$$

Time smo dobili $y = L^{-1}b$ pa sada rješavamo sustav $Rx = y$,

$$\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & r_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} ,$$

koji je gornjetrokutasti i rješavamo ga supstitucijama unazad na isti način kao u Gaussovim eliminacijama. Imamo

$$x_n = \frac{y_n}{r_{nn}} \,,$$

$$x_i = \frac{1}{r_{ii}} \left(y_i - \sum_{j=i+1}^{n} r_{ij} x_j \right) \,, \quad i = n-1, \ldots, 1 \,.$$

Dobili smo rješenje $x = R^{-1}y = R^{-1}L^{-1}b$.

Uočimo da ako znamo faktorizaciju matrice A, onda sustav možemo riješiti vrlo jednostavno koristeći ovu metodu pa je ona korisna za rješavanje sustava linearnih jednadžbi kojima je matrica A sustava jednaka, a mijenja se vektor b. Naravno, pitanje je kako faktorizirati matricu A pa pogledajmo na primjeru matrice 3×3

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} .$$

Prema Gaussovim eliminacijama znamo da najprije želimo poništiti elemente a_{21} i a_{31} pa definiramo matricu

$$L^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ \dfrac{-a_{21}}{a_{11}} & 1 & 0 \\ \dfrac{-a_{31}}{a_{11}} & 0 & 1 \end{bmatrix} . \tag{2.2}$$

Sada je

$$L^{(1)}A = A^{(1)} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} \end{bmatrix},$$

odnosno vrijedi $A = (L^{(1)})^{-1}A^{(1)}$, a lako se izračuna da je

$$(L^{(1)})^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ \dfrac{a_{21}}{a_{11}} & 1 & 0 \\ \dfrac{a_{31}}{a_{11}} & 0 & 1 \end{bmatrix}.$$

Na isti način se sada definira

$$L^{(2)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\dfrac{a_{32}^{(1)}}{a_{22}^{(1)}} & 1 \end{bmatrix}, \qquad (L^{(2)})^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \dfrac{a_{32}^{(1)}}{a_{22}^{(1)}} & 1 \end{bmatrix}, \tag{2.3}$$

za koje vrijedi

$$A^{(2)} = L^{(2)}A^{(1)} = L^{(2)}L^{(1)}A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} \\ 0 & 0 & a_{33}^{(2)} \end{bmatrix},$$

iz čega dobijemo

$$A = (L^{(1)})^{-1}(L^{(2)})^{-1}A^{(2)}.$$

Izračunamo produkt

$$L = (L^{(1)})^{-1}(L^{(2)})^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ \dfrac{a_{21}}{a_{11}} & 1 & 0 \\ \dfrac{a_{31}}{a_{11}} & \dfrac{a_{32}^{(1)}}{a_{22}^{(1)}} & 1 \end{bmatrix}$$

i označimo $R := A^{(2)}$, čime dobivamo rastav matrice A na donje i gornjetrokutastu matricu

$$A = LR = \begin{bmatrix} 1 & 0 & 0 \\ \dfrac{a_{21}}{a_{11}} & 1 & 0 \\ \dfrac{a_{31}}{a_{11}} & \dfrac{a_{32}^{(1)}}{a_{22}^{(1)}} & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} \\ 0 & 0 & a_{33}^{(2)} \end{bmatrix}.$$

Ovaj postupak možemo zapravo izvesti unutar same matrice A tako da je R zapisana u gornjem trokutu, a L u strogo donjem trokutu matrice A. Kako su po dijagonali matrice L samo jedinice, njih ne trebamo zapisivati. Tako zapravo dobijemo matricu

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ \frac{a_{21}}{a_{11}} & a_{22}^{(1)} & a_{23}^{(1)} \\ \frac{a_{31}}{a_{11}} & \frac{a_{32}^{(1)}}{a_{22}^{(1)}} & a_{33}^{(2)} \end{bmatrix}$$

što znatno olakšava zapis algoritma.

Ipak, taj postupak ne možemo provesti općenito za sve regularne matrice. Naime, za definiranje matrice $L^{(1)}$ u (2.2) trebalo je vrijediti $a_{11} \neq 0$, a za definiranje $L^{(2)}$ u (2.3) zahtijevali smo da je $a_{22}^{(1)} \neq 0$. Označimo $\alpha_1 := a_{11}$. Kako je $A = (L^{(1)})^{-1}A^{(1)}$, promatranjem glavne 2×2 podmatrice od A, zaključujemo da je

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{a_{21}}{a_{11}} & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ 0 & a_{22}^{(1)} \end{bmatrix}.$$

Sada odredimo determinantu promatrane podmatrice i označimo je s α_2. Imamo

$$\alpha_2 := \det \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = a_{11}a_{22}^{(1)}$$

iz čega zaključujemo da ako je $\alpha_2 \neq 0$, tada je i $a_{22}^{(1)} \neq 0$. Dakle, već iz ovog jednostavnog primjera možemo primijetiti da za provođenje opisanog postupka LR faktorizacije promatramo determinante glavnih podmatrica matrice A, dimenzija $1, 2, \ldots, n-1$, koje moraju biti različite od nule. Pokažimo u sljedećem teoremu da je to zaista dovoljan uvjet.

Teorem 2.2.1. *[2, Teorem 4.3.1] Neka je A kvadratna matrica reda $n \times n$ i neka su determinante glavnih podmatrica $A(1:k, 1:k)$ različite od nule za $k = 1, 2, \ldots, n-1$. Tada postoji donjetrokutasta matrica L s jedinicama na dijagonali i gornjetrokutasta matrica R takva da je $A = LR$.*

Dokaz. Dokaz provodimo indukcijom po dimenziji matrice. Pretpostavimo da je za neki $k \in \{1, 2, \ldots, n-1\}$ uvjet teorema zadovoljen. Pogledajmo kako tada prelazimo

s $A^{(k)}$ na $A^{(k+1)}$. Neka je

$$A^{(k)} = L^{(k)} \cdots L^{(1)} A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1k} & a_{1,k+1} & \cdots & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & \vdots & a_{2,k+1}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \ddots & a_{33}^{(2)} & & \vdots & a_{3,k+1}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & & \ddots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & \cdots & 0 & a_{kk}^{(k-1)} & a_{k,k+1}^{(k-1)} & \cdots & a_{kn}^{(k-1)} \\ 0 & \cdots & & \cdots & 0 & a_{k+1,k+1}^{(k)} & \cdots & a_{k+1,n}^{k} \\ \vdots & & & & \vdots & \vdots & & \vdots \\ 0 & \cdots & & \cdots & 0 & a_{n,k+1}^{(k)} & \cdots & a_{nn}^{(k)} \end{bmatrix}.$$

Znamo da je produkt $(L^{(k)} \cdots L^{(1)})^{-1}$ donjetrokutasta matrica s jedinicama na dijagonali pa je tada

$$\begin{aligned} \det A(1:k+1,\, 1:k+1) &= \det A^{(k)}(1:k+1,\, 1:k+1) \\ &= a_{11} a_{22}^{(1)} a_{33}^{(2)} \cdots a_{kk}^{(k-1)} a_{k+1,k+1}^{(k)} \neq 0\,, \end{aligned}$$

a kako su prema pretpostavci svi a_{11}, $a_{22}^{(1)}$, $a_{33}^{(2)}, \ldots, a_{kk}^{(k-1)}$ različiti od nule, slijedi da je i $a_{k+1,k+1}^{(k)} \neq 0$. Sada se može definirati $L^{(k+1)}$, a time se dobiva i $A^{(k+1)}$. Postupak se nastavlja sve do $k = n-1$ kada se dobije gornjetrokutasta matrica $A^{(n-1)}$.

□

Općenito, za LR faktorizaciju svake regularne matrice vrši se pivotiranje i dobije se rastav

$$PA = LR\,,$$

gdje je P matrica permutacije kao posljedica pivotiranja.

2.2.2 Algoritam

Zapišimo u Pythonu algoritam za LR faktorizaciju bez pivotiranja zadane matrice A te za rješavanje trokutastih sustava ako su A i b ulazni podaci, a rješenja y i x zapisujemo u b.

Algoritam 2.2.2.

```
1  def LR(A):
2      n=len(A)
3      for k in range (0,n-1):            #LR faktorizacija matrice A
```

```
4           for j in range (k+1,n):
5               if A[k][k]!=0:
6                   A[j][k]=A[j][k]/A[k][k]
7               else:
8                   print("Ne postoji LR faktorizacija.")
9                   exit()
10          for j in range (k+1,n):
11              for i in range (k+1,n):
12                  A[i][j]=A[i][j]-A[i][k]*A[k][j]
13      if A[n-1][n-1]==0:
14          print("Matrica je singularna.")
15          exit()
16      return(A)
17  def rj_LR(A,b):
18      n=len(A)
19      for i in range (1,n):                #supstitucije unaprijed za
20          for j in range (0,i):             sustav Ly=b
21              b[i]=b[i]-A[i][j]*b[j]       #y zapisan u b
22      b[n-1]=b[n-1]/A[n-1][n-1]
23      for i in range (n-2,-1,-1):          #supstitucije unazad za
24          for j in range (i+1,n):           sustav Rx=y
25              b[i]=b[i]-A[i][j]*b[j]       #x zapisan u b
26          b[i]=b[i]/A[i][i]
27      return(b)
```

Uočimo da ako već imamo LR faktorizaciju matrice A, složenost rješavanja sustava $Ax = b$ računamo od linije 17 Algoritma 2.2.2, odnosno rješavanja trokutastih sustava. Kod supstitucija unaprijed imamo

$$\sum_{i=1}^{n-1} i$$

oduzimanja i isto toliko množenja (linija 21) pa je to ukupno

$$2\sum_{i=1}^{n-1} i = 2 \cdot \frac{n(n-1)}{2} = n(n-1) = n^2 - n$$

operacija.

Kod supstitucija unazad imamo n^2 operacija kao što smo već izračunali u potpoglavlju 2.1.3 pa je sveukupno za rješavanje trokutastih sustava potrebno

$$n^2 - n + n^2 = 2n^2 - n$$

operacija, odnosno približno $2n^2$. Dakle, ako već imamo određenu LR faktorizaciju matrice, rješavanje sustava ovom metodom je znatno brže od Gaussovih eliminacija.

Napomenimo još jednom da LR faktorizacija regularnih matrica ne mora postojati (vidi Teorem 2.2.1), već se općenito treba koristiti LR faktorizacija s pivotiranjem, koju nismo u radu obradili.

2.2.3 Računanje inverza

Kao što smo već spomenuli, LR faktorizacija korisna je za rješavanje sustava u kojem se matrica sustava A ne mijenja, a mijenja se vektor b. Takva situacija se javlja pri računanju inverza regularne kvadratne matrice A reda n. Opišimo postupak računanja inverza. Polazimo od jednakosti

$$AA^{-1} = I$$

u kojoj su elementi matrice A i matrice I poznati, a elemente matrice A^{-1} trebamo izračunati. Označimo li

$$A^{-1} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix}$$

tada nam jednakost

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

daje n sustava jednadžbi s n nepoznanica, a svim sustavima je matrica sustava jednaka A. Sada inverznu matricu A^{-1} možemo izračunati koristeći Algoritam 2.2.2 tako da se najprije izračuna LR faktorizacija matrice A, a zatim se računa n sustava za svaki od n stupaca inverzne i jedinične matrice. Primijenimo tu metodu na jednom primjeru.

Primjer 2.2.3. *Odredimo inverz matrice*

$$\begin{bmatrix} 2 & 2 & 1 & 1 \\ 2 & 1 & 0 & 1 \\ 3 & 5 & 1 & 1 \\ 2 & 4 & 2 & 1 \end{bmatrix}.$$

Unošenje ulaznih podataka i poziv funkcije iz Algoritma 2.2.2 vršimo na sljedeći način

```
A=[[2,2,1,1],[2,1,0,1],[3,5,1,1],[2,4,2,1]]
A=LR(A)
n=len(A)
for i in range (0,n):
    b=[0 for i in range (0,n)]
    b[i]=1
    c=rj_LR(A,b)
    print(c)
```

čime dobivamo stupce inverzne matrice A^{-1}

```
[5.0, -2.0, 3.0, -7.999999999999999]
[-2.9999999999999996, 1.0, -2.0, 5.999999999999999]
[1.0, -0.0, -0.0, -2.0]
[-3.0, 1.0, -1.0, 5.0].
```

Dakle, inverzna matrica matrice A je matrica

$$A^{-1} = \begin{bmatrix} 5 & -3 & 1 & -3 \\ -2 & 1 & 0 & 1 \\ 3 & -2 & 0 & -1 \\ -8 & 6 & -2 & 5 \end{bmatrix}.$$

Poglavlje 3

Numeričko rješavanje linearnih sustava: iterativne metode

Ovo poglavlje temelji se na literaturi [2], [5], [6] i [7].

3.1 Općenito o iterativnim metodama

Navedene direktne metode ne garantiraju uvijek točnost rješenja (zbog zaokruživanja rezultata) te su vrlo složene za veće sustave. Zbog toga se umjesto njih često koriste iterativne metode kojima se želi dobiti dovoljno dobra aproksimacija $\tilde{x} \approx A^{-1}b$ rješenja Cramerovog sustava $Ax = b$. Metoda započinje s početnim vektorom $x^{(0)}$, a zatim se konstruira niz vektora $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$, koji konvergira k rješenju linearnog sustava x. Temelj iterativnih metoda je jednostavna formula za računanje $x^{(k)}$, $k \in \mathbb{N}$.

Promotrimo zadani linearni sustav $Ax = b$, gdje je A regularna matrica. Uočimo da sustav možemo ekvivalentno zapisati u obliku

$$x = (I - A)x + b\,. \tag{3.1}$$

Time motiviramo uvođenje iteracije

$$x^{(k+1)} = (I - A)x^{(k)} + b\,, \quad k \in \mathbb{N}\,, \tag{3.2}$$

pri čemu je početna iteracija $x^{(0)}$ zadana. Ukoliko je niz $\left(x^{(k)}\right)_{k\in\mathbb{N}}$ konvergentan te s $\tilde{x}$ označimo pripadni limes, iz prethodne jednakosti (3.2) na limesu dobivamo

$$\tilde{x} = (I - A)\tilde{x} + b\,,$$

iz čega slijedi $A\tilde{x} = b$. Dakle, $\tilde{x}$ je jedinstveno rješenje sustava $Ax = b$. Time je preostalo dati dovoljne uvjete na ulazne podatke A i b koji će osiguravati konvergenciju iteracija $\left(x^{(k)}\right)_{k\in\mathbb{N}}$.

Sve iterativne metode su bazirane na prikazanoj ideji, s tim da je formula kojom su dane iteracije općenito složenija od (3.2). Naime, sustav $Ax = b$ zapišemo u ekvivalentnom obliku

$$x = Mx + c\,, \tag{3.3}$$

pri čemu je M kvadratna matrica i c vektor. Za $M = I - A$ i $c = b$ dobivamo upravo (3.1), ali to općenito nije slučaj, što će biti prikazano u nastavku. Sada konačno rekurzivno definiramo niz iteracija s

$$x^{(k+1)} = Mx^{(k)} + c\,, \quad k \in \mathbb{N}\,, \tag{3.4}$$

uz zadanu početnu iteraciju $x^{(0)}$. U sljedećem najprije dajemo pomoćnu lemu, a zatim i dovoljan uvjet za konvergenciju niza $\left(x^{(k)}\right)_{k\in\mathbb{N}}$, danog s (3.4), k rješenju početnog sustava.

Lema 3.1.1. *Ako je M kvadratna matrica za koju je $\|M\| < 1$ u proizvoljnoj matričnoj normi $\|\cdot\|$, tada je $I - M$ regularna matrica.*

Dokaz. Pokažimo da je $(I - M)^{-1} = \sum_{l=0}^{\infty} M^l$. Najprije ćemo opravdati beskonačnu sumu na desnoj strani jednakosti, odnosno pokazati da pripadni red konvergira. Promotrimo parcijalne sume

$$S_k = \sum_{l=0}^{k} M^l\,.$$

Uočimo da one čine Cauchyjev niz jer je za neki $m > k$

$$\|S_k - S_m\| \leqslant \sum_{l=k+1}^{m} \|M^l\|\,. \tag{3.5}$$

Također vrijedi $\|M^l\| \leqslant \|M\|^l$ pa u (3.5) dobivamo

$$\begin{aligned}\|S_k - S_m\| \leqslant \sum_{l=k+1}^{m} \|M\|^l &= \|M\|^{k+1}\left(1 + \|M\| + \|M\|^2 + \ldots + \|M\|^{m-k-1}\right)\\ &= \|M\|^{k+1}\left(\frac{1 - \|M\|^{m-k}}{1 - \|M\|}\right).\end{aligned}$$

Kako je u pretpostavci teorema $\|M\| < 1$, tada za $m, k \to \infty$ imamo $\|M\|^{k+1} \to 0$. Također je

$$\frac{1 - \|M\|^{m-k}}{1 - \|M\|} \leqslant \frac{1}{1 - \|M\|}$$

pa sada dobijemo da i $\|S_k - S_m\| \to 0$. Time smo pokazali da je niz parcijalnih suma (S_k) Cauchyjev pa je onda i konvergentan, po Teoremu 1.2.5. Neka taj niz konvergira k S.

Iz

$$MS_k = \sum_{l=0}^{k} M^{l+1} = \sum_{n=1}^{k+1} M^n = S_{k+1} - I$$

dobijemo da je

$$MS_k + I = S_{k+1}\,. \tag{3.6}$$

Primjenom limesa na (3.6) dobije se

$$MS + I = S \quad \Longrightarrow \quad S - MS = I \quad \Longrightarrow \quad (I - M)S = I,$$

odnosno

$$S = (I - M)^{-1}$$

čime smo pokazali da je matrica $I - M$ lijevi inverz od S što je dovoljno da bi ona bila regularna. □

Teorem 3.1.2. *Ako je $\|M\| < 1$, pri čemu je $\|\cdot\|$ proizvoljna matrična norma, tada za svaki početni vektor $x^{(0)}$ niz iteracija $\left(x^{(k)}\right)_{k\in\mathbb{N}}$ dan relacijom (3.4) konvergira prema rješenju $x = (I-M)^{-1}c$ Cramerovog sustava $Ax = b$, pri čemu je $A^{-1}b = (I-M)^{-1}c$.*

Dokaz. Neka je x jedinstveno rješenje sustava $Ax = b$, odnosno

$$x = A^{-1}b = (I - M)^{-1}c\,. \tag{3.7}$$

Iz (3.7) tada slijedi

$$(I - M)x = c \quad \Rightarrow \quad x - Mx = c \quad \Rightarrow \quad x = Mx + c\,.$$

Pogledajmo sada razliku

$$x^{(k+1)} - x = Mx^{(k)} + c - Mx - c = M(x^{(k)} - x)\,. \tag{3.8}$$

Uzimanjem norme izraza (3.8) i oznake $e^{(i)} := x^{(i)} - x$, $i \in \mathbb{N}_0$, dobije se

$$\|e^{(k+1)}\| \leqslant \|M\|\|e^{(k)}\|\,,$$

pri čemu smo koristili istu oznaku za matričnu i pripadnu vektorsku normu.

Nastavimo li iteracije, imamo

$$\|e^{(k+1)}\| \leqslant \|M\|\|e^{(k)}\| \leqslant \|M\|^2\|e^{(k-1)}\| \leqslant \ldots \leqslant \|M\|^{k+1}\|e^{(0)}\|\,, \tag{3.9}$$

gdje je $e^{(0)} = x^{(0)} - x$, a $x^{(0)}$ je proizvoljno odabrana prva aproksimacija. Po pretpostavci teorema je $\|M\| < 1$ pa je

$$\lim_{k\to\infty} \|M\|^{k+1} = 0\,,$$

a zbog (3.9) imamo

$$\lim_{k\to\infty} \|e^{(k+1)}\| = 0\,,$$

odnosno

$$\lim_{k\to\infty} \|x^{(k+1)} - x\| = 0\,,$$

čime smo pokazali da $x^{(k)}$ konvergira k x. □

3.2 Jacobijeva metoda

3.2.1 Opis metode

Ideja Jacobijeve metode je rastaviti kvadratnu matricu A za koju je $a_{ii} \neq 0$, $i = 1, \ldots, n$, kao $A = D - N$ pri čemu je $D = \operatorname{diag}(a_{11}, \ldots, a_{nn})$. Tada je $N = D - A$, odnosno N je izvandijagonalni dio od $-A$. Početni sustav $Ax = b$ u tom slučaju možemo zapisati kao

$$(D - N)x = b \quad \Rightarrow \quad Dx - Nx = b \quad \Rightarrow \quad Dx = Nx + b\,.$$

Kako je D regularna matrica konačno dobivamo

$$x = D^{-1}Nx + D^{-1}b\,,$$

pa je Jacobijeva metoda dana relacijom

$$x^{(k+1)} = D^{-1}Nx^{(k)} + D^{-1}b\,, \quad k \in \mathbb{N}\,.$$

Dakle, u ovom slučaju je $M = D^{-1}N$ pa prema Teoremu 3.1.2 metoda konvergira ako je $\|D^{-1}N\| < 1$ u proizvoljnoj matričnoj normi.

Zapišimo sada eksplicitno Jacobijeve iteracije po elementima. Promotrimo na primjeru sustava 3×3

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3\,, \end{aligned}$$

za koji je $a_{ii} \neq 0\,,\ i = 1\,, 2\,, 3$. Iz svake od tih jednadžbi imamo

$$\begin{aligned} x_1 &= \frac{1}{a_{11}}\left(b_1 - a_{12}x_2 - a_{13}x_3\right) \\ x_2 &= \frac{1}{a_{22}}\left(b_2 - a_{21}x_1 - a_{23}x_3\right) \\ x_3 &= \frac{1}{a_{33}}\left(b_3 - a_{31}x_1 - a_{32}x_2\right)\,. \end{aligned}$$

Sada za neku početnu aproksimaciju

$$x^{(0)} = \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \end{bmatrix}$$

računamo sljedeću $x^{(1)}$ s

$$\begin{aligned} x_1^{(1)} &= \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^{(0)} - a_{13}x_3^{(0)}\right) \\ x_2^{(1)} &= \frac{1}{a_{22}}\left(b_2 - a_{21}x_1^{(0)} - a_{23}x_3^{(0)}\right) \\ x_3^{(1)} &= \frac{1}{a_{33}}\left(b_3 - a_{31}x_1^{(0)} - a_{32}x_2^{(0)}\right)\,. \end{aligned} \tag{3.10}$$

Vidimo da općenito, za sustav $n \times n$, komponente vektora $x^{(k+1)}$ možemo računati pomoću relacije

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j^{(k)}\right), \qquad i = 1, \ldots, n\,.$$

3.2.2 Algoritam

Ako su ulazni podaci matrica A, čiji dijagonalni elementi nisu nula, vektor b, zatim vektor x kao početna aproksimacija te k_{max} kao broj koraka, zapišimo u Pythonu algoritam Jacobijeve metode.

Algoritam 3.2.1.

```
def Jacobi(A,b,x,k_max):
    n=len(b)
    for k in range (0,k_max+1):
        xstari=x
        for i in range (0,n):
            sum=0
            for j in range (0,n):
                if j!=i:
                    sum=sum+A[i][j]*xstari[j]
            x[i]=(1/A[i][i])*(b[i]-sum)
        print(x)
    return(x)
```

3.3 Gauss-Seidelova metoda

3.3.1 Opis metode

Za razliku od Jacobijeve metode u kojoj se koriste aproksimacije x iz prethodne iteracije, u Gauss-Seidelovoj metodi se uvijek koriste najnovije izračunate aproksimacije. Ako pogledamo na istom primjeru 3×3 sustava, onda umjesto (3.10), sada imamo

$$\begin{aligned} x_1^{(1)} &= \frac{1}{a_{11}} \left(b_1 - a_{12}x_2^{(0)} - a_{13}x_3^{(0)} \right) \\ x_2^{(1)} &= \frac{1}{a_{22}} \left(b_2 - a_{21}x_1^{(1)} - a_{23}x_3^{(0)} \right) \\ x_3^{(1)} &= \frac{1}{a_{33}} \left(b_3 - a_{31}x_1^{(1)} - a_{32}x_2^{(1)} \right) , \end{aligned}$$

odnosno općenito $x^{(k+1)}$ računamo pomoću formule

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} \right) , \qquad i = 1, \ldots, n \, .$$

Kao i za Jacobijevu metodu, pogledajmo rastav matrice A i zapis metode pomoću tih matrica. I dalje ćemo proučavati sustav 3×3, no zapišimo komponente vektora $x^{(k+1)}$ na sljedeći način

$$\begin{array}{lllll} a_{11}x_1^{(k+1)} & & & = & b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} \\ a_{21}x_1^{(k+1)} + a_{22}x_2^{(k+1)} & & & = & b_2 - a_{23}x_3^{(k)} \\ a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)} + a_{33}x_3^{(k+1)} & & & = & b_3 \end{array} \, .$$

Prevedemo li to u matrični oblik imamo

$$\begin{bmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ x_3^{(k+1)} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} + \begin{bmatrix} 0 & -a_{12} & -a_{13} \\ 0 & 0 & -a_{23} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \\ x_3^{(k)} \end{bmatrix}. \tag{3.11}$$

Označimo sada

$$L := \begin{bmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \quad U := \begin{bmatrix} 0 & -a_{12} & -a_{13} \\ 0 & 0 & -a_{23} \\ 0 & 0 & 0 \end{bmatrix}$$

i uočimo da vrijedi $A = L - U$, odnosno to je rastav matrice A sustava koji nam treba za Gauss-Seidelovu metodu. Uz te oznake i rastav (3.11), uočimo da iteracije možemo zapisati kao

$$x^{(k+1)} = L^{-1}(b + Ux^{(k)})$$

iz čega vidimo da je $M = L^{-1}U$ pa prema Teoremu 3.1.2 metoda konvergira ako je $\|L^{-1}U\| < 1$, u nekoj matričnoj normi $\|\cdot\|$.

3.3.2 Algoritam

Zapišimo u Pythonu algoritam Gauss-Seidelove metode s ulaznim podacima matricom A, kojoj elementi na dijagonali nisu nula, vektorom b, vektorom x kao početnom aproksimacijom te k_{max} kao brojem koraka.

Algoritam 3.3.1.

```
1  def GS(A,b,x,k_max):
2      n=len(b)
3      for k in range (0,k_max +1):
4          for i in range (0,n):
5              sum=0
6              for j in range (0,n):
7                  if j!=i:
8                      sum=sum+A[i][j]*x[j]
9              x[i]=(1/A[i][i])*(b[i]-sum)
10         print(x)
11     return(x)
```

3.4 Uvjeti konvergencije

U Teoremu 3.1.2 dali smo uvjet konvergencije iterativnih metoda koji je često teško provjeriti pa ćemo promotriti što je dovoljno da vrijedi za matricu sustava kako bi taj dani uvjet bio zadovoljen.

Definicija 3.4.1. *Za kvadratnu matricu A tipa $n \times n$ kažemo da je:*
a) strogo dijagonalno dominantna po recima ako vrijedi

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad \forall i \in \{1, \ldots, n\}.$$

b) jako dijagonalno dominantna po recima ako vrijedi

$$|a_{ii}| \geqslant \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad \forall i \in \{1, \ldots, n\},$$

te stroga nejednakost vrijedi barem za jedan i.

Teorem 3.4.2. *Ako je A strogo dijagonalno dominantna matrica po recima, onda Jacobijeva metoda konvergira prema rješenju sustava $Ax = b$ za svaku početnu iteraciju $x^{(0)}$.*

Dokaz. Kako bi dokazali da Jacobijeva metoda konvergira, dovoljno je dokazati da je $\|D^{-1}N\|_\infty < 1$ za $A = D - N$, gdje je $D = \operatorname{diag}(a_{11}, \ldots, a_{nn})$, a N je izvandijagonalni dio od $-A$. Za matricu

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

tada imamo

$$D^{-1} = \begin{bmatrix} \frac{1}{a_{11}} & 0 & \cdots & 0 \\ 0 & \frac{1}{a_{22}} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & \frac{1}{a_{nn}} \end{bmatrix}, \qquad N = \begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & 0 & \cdots & -a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & 0 \end{bmatrix},$$

odnosno

$$D^{-1}N = \begin{bmatrix} 0 & \frac{-a_{12}}{a_{11}} & \cdots & \frac{-a_{1n}}{a_{11}} \\ \frac{-a_{21}}{a_{22}} & 0 & \cdots & \frac{-a_{2n}}{a_{22}} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{-a_{n1}}{a_{nn}} & \frac{-a_{n2}}{a_{nn}} & \cdots & 0 \end{bmatrix}.$$

Prema definiciji matrične beskonačne norme sada imamo

$$\|D^{-1}N\|_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{n} |(D^{-1}N)_{ij}| = \max_{i=1,\ldots,n} \frac{1}{|a_{ii}|} \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|\,. \tag{3.12}$$

Iz pretpostavke teorema je

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|, \quad i = 1, \ldots, n\,,$$

pa kad to uvrstimo u (3.12) dobijemo

$$\|D^{-1}N\|_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{n} |a_{ij}| = \max_{i=1,\ldots,n} \frac{1}{|a_{ii}|} \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| < \max_{i=1,\ldots,n} \frac{1}{|a_{ii}|} |a_{ii}| = 1$$

čime smo pokazali traženu tvrdnju. □

Prezentirat ćemo još jedan rezultat konvergencije iterativnih metoda uz drugačije pretpostavke od onih u prethodnom teoremu, a za koji nam je potrebna sljedeća definicija.

Definicija 3.4.3. *Za kvadratnu matricu A kažemo da je reducibilna ako postoji netrivijalna particija $\{1, \ldots, n\} = I \cup J$ takva da za $(i, j) \in I \times J$ vrijedi $a_{ij} = 0$. U suprotnom je matrica A ireducibilna.*

Primjer jedne reducibilne matrice je blok matrica

$$\begin{bmatrix} A & B \\ 0 & D \end{bmatrix},$$

gdje je blok element $(2, 1)$ nul-matrica, A je matrica dimenzije $i \times i$, B dimenzije $i \times j$, a D dimenzije $j \times i$.

Sada dajemo iskaz najavljenog teorema koji nećemo dokazivati.

Teorem 3.4.4. *[5, Propozicija 9.3.1] Ako je A ireducibilna i jako dijagonalno dominantna matrica po recima, onda Jacobijeva i Gauss-Seidelova metoda konvergiraju prema rješenju sustava $Ax = b$.*

Prethodni teorem nam je bitan u ovom radu jer iz njega slijedi konvergencija Jacobijeve i Gauss-Seidelove metode u slučaju kada je matrica sustava tridijagonalna. Više o tome u poglavlju 3.6.

3.5 Osnovni primjeri

Primjer 3.5.1. *Riješimo sustav $Ax = b$ gdje je*

$$A = \begin{bmatrix} 6 & -2 & 1 \\ -2 & 7 & 2 \\ 1 & 2 & -5 \end{bmatrix}, \quad b = \begin{bmatrix} 11 \\ 5 \\ -1 \end{bmatrix},$$

pomoću Jacobijeve metode, uz početnu iteraciju

$$x^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Jacobijevu metodu provodimo za 20 koraka. Unošenje ulaznih podataka i poziv funkcije vršimo na sljedeći način

```
A=[[6,-2,1],[-2,7,2],[1,2,-5]]
b=[11,5,-1]
x=[0,0,0]
k_max=20
x=Jacobi(A,b,x,k_max)
print(x)
```

pri čemu je funkcija Jacobi dana u Algoritmu 3.2.1. Program ispisuje sljedeće iteracije

```
[1.8333333333333333, 1.238095238095238, 1.0619047619047762]
[2.069047619047619, 1.0020408163265306, 1.014625850340136]
[1.9982426303854874, 0.9953190800129575, 0.9977761580822805]
[1.9988103336572725, 1.000295478735712, 0.9998802582257392]
[2.0001184498742806, 1.000068054756726, 1.0000509118775465]
[2.000014199605984, 0.9999895107795536, 0.9999986442330183]
[1.9999967295543482, 0.9999994529489513, 0.9999991270904502]
```

```
[1.9999999631345755, 1.00000023886975, 1.0000000881748l5]
[2.0000000649274474, 0.9999999933578949, 1.00000001032864 75]
[1.9999999960645236, 0.9999999959245359, 0.99999999758271 91]
[1.9999999990443922, 1.0000000004176208, 0.99999999997592 68]
[2.0000000001432188, 1.0000000000477975, 1.00000000004776 27]
[2.0000000000079 72, 0.9999999999886311, 0.99999999999704 68]
[1.9999999999967024, 0.9999999999999015, 0.99999999999930 12]
[2.0000000000000835, 1.0000000000002236, 1.00000000000010 64]
[2.000000000000057, 0.9999999999999858, 1.00000000000000 58]
[1.9999999999999942, 0.9999999999999967, 0.99999999999999 76]
[1.9999999999999993, 1.0000000000000004, 1.0]
[2.0, 1.0, 1.0]
[2.0, 1.0, 1.0]
[2.0, 1.0, 1.0]
[2.0, 1.0, 1.0].
```

Primijetimo kako je zadana matrica A strogo dijagonalno dominantna pa očekivano metoda konevrgira k rješenju sustava i već za $k_{max} = 18$ daje egzaktno rješenje

$$x = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}.$$

Primjer 3.5.2. *Riješimo sustav $Ax = b$ gdje je*

$$A = \begin{bmatrix} 1 & 2 & -1 & 1 \\ 2 & 5 & -1 & 2 \\ 3 & -1 & 2 & 1 \\ 1 & -1 & 3 & -5 \end{bmatrix}, \quad b = \begin{bmatrix} -1 \\ -2 \\ 5 \\ 6 \end{bmatrix},$$

pomoću Jacobijeve metode, uz početnu iteraciju

$$x^{(0)} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Za ovaj primjer zaokružiti ćemo dobivene elemente vektora $x^{(k)}$ na 2 decimale tako da u liniji 10 Algoritma 3.2.1 pišemo

```
x[i]=round((1/A[i][i])*(b[i]-sum),2).
```

Metodu provodimo za 10 koraka. U nastavku vršimo unošenje ulaznih podataka i poziv funkcije na sljedeći način

```
A=[[1,2,-1,1],[2,5,-1,2],[3,-1,-2,1],[1,-1,3,-5]]
b=[-1,-2,5,6]
x=[1,1,1,1]
k_max=10
x=Jacobi(A,b,x,k_max)
print(x)
```

pri čemu je funkcija Jacobi dana u Algoritmu 3.2.1. Program ispisuje sljedeće iteracije

```
[-3.0, 0.6, -6.8, -6.0]
[-3.0, 1.84, -10.92, -8.72]
[-6.88, 3.66, -19.01, -14.71]
[-12.62, 6.73, -32.15, -24.36]
[-22.25, 11.81, -53.96, -40.39]
[-38.19, 20.24, -90.1, -66.95]
[-64.63, 34.21, -150.03, -110.99]
[-108.46, 57.37, -249.37, -183.99]
[-181.12, 95.77, -414.06, -305.01]
[-301.59, 159.43, -687.11, -505.67]
[-501.3, 264.97, -1139.77, -838.32]
[-501.3, 264.97, -1139.77, -838.32].
```

Egzaktno rješenje sustava je vektor

$$x^{(0)} = \begin{bmatrix} 2 \\ -1 \\ 1 \\ 0 \end{bmatrix}.$$

Uočimo da matrica A nije strogo (niti jako) dijagonalno dominatna. Provjerimo u sljedećoj tablici koja je pogreška u odnosu na egzaktno rješenje za određene iteracije.

k	$x^{(k)}$	$\|x^{(k)} - x\|$
0	$(-3.0, 0.6, -6.8, -6.0)$	7.8
5	$(-38.19, 20.24, -90.1, -66.95)$	91.1
10	$(-501.3, 264.97, -1139.77, -838.32)$	1140.77

Možemo vidjeti da se pogreška povećava, dakle metoda ne konvergira k egzaktnom rješenju zadanog sustava.

3.6 Numeričko rješavanje diferencijalne jednadžbe

Opišimo primjer obične diferencijalne jednadžbe dan u [2].

Zadan je rubni problem

$$-\frac{d^2}{dx^2}u(x) = f(x), \qquad 0 < x < 1 \tag{3.13}$$

$$u(0) = u(1) = 0. \tag{3.14}$$

Tražimo rješenje u problema. Opišimo metodu kojom dolazimo do aproksimacije rješenja na skupu od konačno mnogo točaka

$$0 = x_0 < x_1 < x_2 < \ldots < x_n < x_{n+1} = 1\,, \qquad n \in \mathbb{N}\,,$$

iz segmenta $[0, 1]$.

Definiramo korak

$$h = \frac{1}{n+1}$$

i čvorove

$$x_i = ih\,, \qquad i = 1, \ldots, n+1\,,$$

te označimo tražene vrijednosti

$$u_i = u(x_i)\,, \qquad i = 0, \ldots, n+1\,.$$

Polazeći od funkcije u koja zadovoljava (3.13) i (3.14), u svakom čvoru želimo zamijeniti diferencijalnu jednadžbu odgovarajućom algebarskom jednadžbom kako bismo dobili i riješili sustav algebarskih jednadžbi. Radi toga mijenjamo derivacije s odgovarajućim algebarskim aproksimacijama. Prema Taylorovom teoremu srednje vrijednosti, za aproksimaciju druge derivacije imamo

$$u_{i+1} = u(x_i + h) = u_i + u_i'h + \frac{u_i''}{2!}h^2 + \frac{u_i'''}{3!}h^3 + \frac{u^{(4)}(x_i + \alpha_i)}{4!}h^4\,, \tag{3.15}$$

$$u_{i-1} = u(x_i - h) = u_i - u_i'h + \frac{u_i''}{2!}h^2 - \frac{u_i'''}{3!}h^3 + \frac{u^{(4)}(x_i + \beta_i)}{4!}h^4\,, \tag{3.16}$$

gdje je $\alpha_i \in (x_i\,, x_i + h)$ i $\beta_i \in (x_i - h\,, x_i)$. Iz početnog uvjeta (3.14) slijedi da je

$$u_0 = u_{n+1} = 0\,.$$

Sada zbrojimo jednadžbe (3.15) i (3.16) i dobijemo

$$u_{i+1} + u_{i-1} = 2u_i + u_i'' h^2 + (u^{(4)}(x_i + \alpha_i) + u^{(4)}(x_i + \beta_i))\frac{h^4}{24}, \quad i = 1, \ldots, n\,.$$

Označimo $\epsilon_i = -(u^{(4)}(x_i + \alpha_i) + u^{(4)}(x_i + \beta_i))\frac{h^4}{24}$ i tada je

$$-u_i'' = \frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} - \epsilon_i, \quad i = 1, \ldots, n\,.$$

Kako je, zbog početnog uvjeta (3.13), $-u_i'' = f_i$, slijedi

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} = f_i + \epsilon_i, \quad i = 1, \ldots, n\,.$$

Sada iskoristimo zadane rubne uvjete (3.14) da pojednostavimo prvu i posljednju jednadžbu.
Za $i = 1$ imamo

$$\begin{aligned} \frac{-u_0 + 2u_1 - u_2}{h^2} &= f_1 + \epsilon_1 \\ 2u_1 - u_2 &= f_1 h^2 + \epsilon_1 h^2\,. \end{aligned}$$

Za $i = n$ imamo

$$\begin{aligned} \frac{-u_{n-1} + 2u_n - u_{n+1}}{h^2} &= f_n + \epsilon_n \\ -u_{n-1} + 2u_n &= f_n h^2 + \epsilon_n h^2\,. \end{aligned}$$

Uočimo da tako dobivamo sustav od n jednadžbi s n nepoznanica koji možemo zapisati matrično na sljedeći način

$$\begin{bmatrix} 2 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & \cdots & 0 & 0 \\ 0 & -1 & 2 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2 & -1 \\ 0 & 0 & 0 & \cdots & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = h^2 \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{n-1} \\ f_n \end{bmatrix} + h^2 \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_{n-1} \\ \epsilon_1 \end{bmatrix}, \tag{3.17}$$

odnosno uz pripadajuće oznake sustav

$$T_n u = h^2 f + h^2 \epsilon\,.$$

Matrica T_n je gore eksplicitno dana, dok vektor f dobivamo evaluiranjem desne strane jednakosti (3.13) u čvorovima. Vektor ϵ a priori ne znamo pa njega izostavljamo, što opravdavamo činjenicom da je ϵ reda veličine h^4, odnosno za mali h se radi o vrlo maloj veličini pa se nadamo da neće bitno utjecati na točnost aproksimacije. Time konačno promatramo sustav

$$T_n\tilde{u} = h^2 f\,, \tag{3.18}$$

gdje $\tilde{u}$ aproksimira rješenje u, odnosno vrijednost rješenja u čvorovima x_i, $i = 1, \ldots, n$.

U nastavku pokazujemo da je sustav (3.18) pogodan za primjenu Jacobijeve i Gauss-Seidelove metode.

Definicija 3.6.1. *Za kvadratnu matricu A kažemo da je tridijagonalna ako je*

$$a_{ij} = 0 \quad za \quad |j - i| \geqslant 2\,,$$

odnosno ako je oblika

$$\begin{bmatrix} * & * & 0 & \cdots & 0 \\ * & \ddots & \ddots & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & * \\ 0 & \cdots & 0 & * & * \end{bmatrix}.$$

Lema 3.6.2. *[5, Zadatak 2.9.15.(a)] Kvadratna matrica A je ireducibilna ako i samo ako za svaki par (i, j), $1 \leqslant i\,, j \leqslant n$, $i \neq j$, postoji konačan niz indeksa $i = l_1, \ldots, l_r = j$ takav da je $a_{l_p, l_{p+1}} \neq 0$, $p = 1, \ldots, r-1$.*

Korolar 3.6.3. *Tridijagonalna matrica A za koju nijedan od elemenata $a_{i,i+1}$ i $a_{i+1,i}$ nije nula je ireducibilna.*

Dokaz. Trebamo pokazati da ovi uvjeti zadovoljavaju pretpostavku Leme 3.6.2. Odaberimo prozvoljne i, j. Razlikujemo dva slučaja, kada je $j > i$ i kada je $i > j$.
Za $j > i$ su elementi

$$a_{i,i+1}\,,\ a_{i+1,i+2}\,,\ a_{i+2,i+3}\,,\ \ldots\,,\ a_{j-1,j}$$

različiti od 0 i time zadovoljavaju uvjete Leme 3.6.2.
Za $i > j$ su elementi

$$a_{i,i-1}\,,\ a_{i-1,i-2}\,,\ a_{i-2,i-3}\,,\ \ldots\,,\ a_{j,j-1}$$

također različiti od 0 i time zadovoljavaju uvjete Leme 3.6.2 čime smo pokazali da je tridijagonalna matrica s navedenim svojstvima ireducibilna. □

Uočimo da je matrica sustava T_n, dobivena u (3.17), tridijagonalna, a prema Korolaru 3.6.3 je ona ireducibilna. Matrica je također jako dijagonalno dominantna po recima jer za svaki $i = 1, \ldots, n$ vrijedi

$$2 = |a_{ii}| \geqslant \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| = 1 + 1 + 0 + \ldots + 0\,,$$

a za $i = 1$ i $i = n$ vrijedi stroga nejednakost, odnosno

$$2 = |a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| = 1 + 0 + 0 + \ldots + 0\,.$$

Prema tome, Jacobijeva i Gauss-Seidelova metoda za matricu T_n konvergiraju prema rješenju sustava neovisno o izboru početne iteracije. Sada prezentiramo primjenu Gaussovih eliminacija i Gauss-Seidelove metode u rješavanju sustava (3.18) za neke konkretne izbore desne strane f.

Primjer 3.6.4. *Numerički riješimo diferencijalnu jednadžbu*

$$\begin{aligned} -\frac{d^2}{dx^2}u(x) &= \pi^2 \sin(\pi x)\,, \qquad 0 < x < 1 \\ u(0) &= u(1) = 0\,, \end{aligned}$$

za $n = 20$, koristeći metodu Gaussovih eliminacija i usporedimo dobiveno rješenje s egzaktnim rješenjem $u(x) = \sin(\pi x)$.

Kako bismo riješili primjer opisanim postupkom za $n = 20$, moramo definirati korak h te čvorove x_i, $i = 1, \ldots, n+1$. Zatim, za rješavanje sustava (3.18) potrebno je definirati tridijagonalnu matricu sustava T_n i vektor desne strane f. Tada pozivanjem funkcije iz Algoritma 2.1.1 dobijemo aproksimaciju rješenja u. Za određivanje apsolutne pogreške dobivene aproksimacije s obzirom na rješenje jednadžbe, u kodu ćemo zapisati vektor s egzaktnim vrijednostima u_i, $i = 1, \ldots, n$, te promatrati euklidsku normu njihove razlike. Dakle, u Pythonu zapisujemo sljedeći algoritam

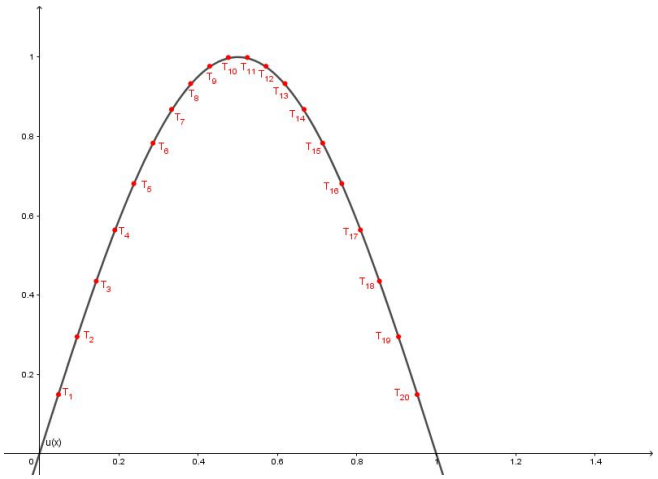
```
1  import math
2
3  n=20
4  h=1/(n+1)
5
6  x=[]                          #tocke x_i unutar intervala [0,1]
```

```
for i in range (0,n):
        x.append((i+1)*h)
print(x)

T=[]                          #definiramo tridijagonalnu matricu T
for i in range (0,n):
    T.append([])
    for j in range (0,n):
        T[i].append(0)
        if i==j:
            T[i][j]=2
        if i==j+1 or i==j-1:
            T[i][j]=-1

f=[]                         #definiramo vektor desne strane sustava
for i in range (0,n):
        f.append((h**2)*(math.pi**2*math.sin(math.pi*x[i])))

u_gauss=gauss(T,f)
print(u_gauss)

def norm(x,y):           #definiramo funkciju euklidske norme
    n=len(x)
    sum=0.0
    for i in range (0,n):
        sum+=(x[i]-y[i])**2
    return(math.sqrt(sum))

u_rj=[]                          #egzaktno rješenje jednadžbe
for i in range (0,n):
        u_rj.append(math.sin(math.pi*x[i]))

print(norm(u_gauss,u_rj))        #apsolutna pogreska metode
```

koji nam u ispisu daje 20 točaka označenih na Slici 3.1 te pogrešku

0.006050074128593047.

Iz dobivenih rezultata možemo vidjeti da smo dobili jako dobru aproksimaciju rješenja, s pogreškom tek na trećoj decimali.



Slika 3.1: Usporedba točnog rješenja jednadžbe i dobivenih točaka

Pogledajmo u nastavku kojom brzinom metoda Gaussovih eliminacija dolazi do rješenja i kako se ona mijenja s povećanjem n.

Primjer 3.6.5. *Zadana je diferencijalna jednadžba*

$$\begin{aligned} -\frac{d^2}{dx^2}u(x) &= 2\,, \qquad 0 < x < 1 \\ u(0) &= u(1) = 0\,. \end{aligned}$$

Izračunajmo brzinu izvršenja metode Gaussovih eliminacija za $n = 10, 100, 500, 1000$.

Za mjerenje brzine izvšenja, potrebno je u Pythonu na početku algoritma dodati naredbu

```
import time.
```

Nakon toga na isti način kao u Primjeru 3.6.4 definiramo n, h, x, T_n, *dok vektor* f, *poziv funkcije iz Algoritma 2.1.1 te brzinu izvršenja računamo koristeći sljedeći kod*

```
f=[]                              #definiramo vektor desne strane sustava
for i in range (0,n):
        f.append(h**2*2)

pocetno_vrijeme = time.time()
u_gauss=gauss(T,f)
ukupno_vrijeme = time.time() - start_time
print(ukupno_vrijeme).
```

U sljedećoj tablici prikazani su rezultati za različite n izmjereni na računalu s procesorom Intel(R) Pentium(R) CPU 2020M, 2.40 GHz i 4 GB RAM-a.

n	t
50	$0.025000572204589844s$
100	$0.1795027256011963s$
500	$20.411316633224487s$
1000	$177.8903844356537s$

Uočimo da kada n povećamo primjerice 10 puta, vrijeme izvršavanja se povećalo približno 1000 puta što je u skladu s izračunatom kubnom složenosti metode Gaussovih eliminacija.

Napomenimo da je rješenje diferencijalne jednadžbe u prethodnom primjeru $u(x) = -x(x-1)$, za koju su treća i četvrta derivacija u relacijama (3.15) i (3.16) jednake nuli, pa je i $\epsilon = 0$, tako da se opisanim postupkom ne dobiva aproksimacija već točno rješenje u. Iskoristiti ćemo istu funkciju u sljedećem primjeru u kojem ćemo usporediti metodu Gaussovih eliminacija i Gauss-Seidelovu metodu.

Primjer 3.6.6. *Zadana je diferencijalna jednadžba*

$$\begin{aligned} -\frac{d^2}{dx^2}u(x) &= 2\,, \qquad 0 < x < 1 \\ u(0) &= u(1) = 0\,. \end{aligned}$$

Izračunajmo pogrešku te brzinu izvršenja za metodu Gaussovih eliminacija i Gauss-Seidelovu metodu kada je $n = 20$ i $n = 500$, dok je broj iteracija u Gauss-Seidelovoj metodi fiksiran na $k_{max} = 150$.

Za izračun vremena izvršenja i pogreške obiju metoda pišemo u Pythonu sljedeći kod

```
1  import time
2  import math
3
4  n=20
5  h=1/(n+1)
6
7  x=[]                          #tocke x_i unutar intervala [0,1]
8  for i in range (0,n):
9          x.append((i+1)*h)
10
```

```
T=[]                              #definiramo tridijagonalnu matricu T
for i in range (0,n):
    T.append([])
    for j in range (0,n):
        T[i].append(0)
        if i==j:
            T[i][j]=2
        if i==j+1 or i==j-1:
            T[i][j]=-1

f=[]                             #definiramo vektor desne strane sustava
for i in range (0,n):
        f.append(h**2*2)

k_max=150        #broj iteracija u GS metodi
u_poc=[]        #pocetna iteracija
for i in range (0,n):
        u_poc.append(0)

pocetno_vrijeme = time.time()            #računamo rješenje uz
u_GS=GS(T,f,u_poc,k_max)                  vrijeme izvršavanja
ukupno_vrijeme = time.time() - pocetno_vrijeme
print(ukupno_vrijeme)

pocetno_vrijeme = time.time()
u_gauss=gauss(T,f)
ukupno_vrijeme = time.time() - pocetno_vrijeme
print(ukupno_vrijeme)

def norm(x,y):          #definiramo funkciju euklidske norme
    n=len(x)
    sum=0.0
    for i in range (0,n):
        sum+=(x[i]-y[i])**2
    return(math.sqrt(sum))

u_rj=[]                           #egzaktno rješenje jednadžbe
for i in range (0,n):
         u_rj.append(-x[i]*(x[i]-1))
```

```
50
51 print(norm(u_GS,u_rj))          #apsolutna pogreska svake metode
52 print(norm(u_gauss,u_rj)),
```

u kojem pozivamo funkcije definirane u Algoritmu 2.1.1 i Algoritmu 3.3.1. Dobivene rezultate za $n = 20$ *i* $n = 500$*, pri čemu smo ih zaokružili na 3 decimale, prikazujemo u sljedećoj tablici.*

n	t - *GS*	*pogreška - GS*	t - *Gauss*	*pogreška - Gauss*
20	$0.026s$	0.028	$0.001s$	$2.869e-16$
500	$15.094s$	4.062	$20.570s$	$2.904e-13$

Možemo vidjeti da je za male n metoda Gaussovih eliminacija brža i točnija. S povećanjem n, brzina izvođenja te metode se kubno povećava, no točnost ostaje dovoljno dobra za razliku od Gauss-Seidelove metode koja je za veće n puno brža, ali je pogreška u odnosu na egzaktni rezultat puno veća.

U prethodnom primjeru smo za Gauss-Seidelovu metodu fiksirali k_{max} i mijenjali n. Pogledajmo sada na primjeru iste diferencijalne jednadžbe što se događa ako fiksiramo n i mijenjamo k_{max}.

Primjer 3.6.7. *Zadana je diferencijalna jednadžba*

$$\begin{aligned} -\frac{d^2}{dx^2}u(x) &= 2\,, \qquad 0 < x < 1 \\ u(0) &= u(1) = 0\,. \end{aligned}$$

Računajmo rješenje jednadžbe za $n = 20$*, koristeći Gauss-Seidelovu metodu te pogledajmo njezinu točnost i brzinu za* $k_{max} = 100, 500, 1000, 1500$*.*

U ovom primjeru koristimo isti kod kao i u Primjeru 3.6.6, ali bez dijela koda koji računa Gaussove eliminacije. Rezultate u ovisnosti o k_{max} *dajemo u sljedećoj tablici, s time da smo ih zaokružili na 3 decimale.*

k_{max}	t	*pogreška*
100	$0.015s$	0.087
500	$0.090s$	$1.092e-05$
1000	$0.175s$	$1.446e-10$
1500	$0.260s$	$6.733e-16$

Prema dobivenim rezultatima možemo uočiti da se apsolutna pogreška smanjuje s povećanjem k_{max}*, dakle metoda konvergira prema rješenju jednadžbe. Naravno, vrijeme izvršenja se povećava približno koliko i* k_{max}*, jer je složenost Gauss-Seidelove metode linearna u ovisnosti o* k_{max}*.*

Bibliografija

[1] D. Bakić, *Linearna algebra*, Školska knjiga, Zagreb, 2008.

[2] Z. Drmač, M. Marušić, S. Singer, V. Hari, M. Rogina, S. Singer, *Numerička analiza*, nastavni materijal za kolegij Numerička matematika, Sveučilište u Zagrebu, PMF - Matematički odjel, Zagreb, 2003.
Dostupno na: `https://web.math.pmf.unizg.hr/~singer/num_mat/num_anal.pdf` (stranica posjećena: 20. kolovoza 2018.)

[3] Z. Drmač, M. Marušić, S. Singer, V. Hari, M. Rogina, S. Singer, *Numerička matematika*, nastavni materijal za kolegij Numerička matematika, Sveučilište u Zagrebu, PMF - Matematički odjel, Zagreb, 2008.
Dostupno na: `https://web.math.pmf.unizg.hr/~singer/num_mat/num_mat1.pdf` (stranica posjećena: 20. kolovoza 2018.)

[4] D. Bakić, *Normirani prostori*, nastavni materijal za kolegij Normirani prostori, Sveučilište u Zagrebu, PMF - Matematički odjel, Zagreb, 2017.
Dostupno na: `https://web.math.pmf.unizg.hr/nastava/np/predavanja/np-1718.pdf` (stranica posjećena: 20. kolovoza 2018.)

[5] D. Serre, *Matrices: Theory And Applications*, Springer-Verlag New York, Inc, New York, 2002.

[6] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, Society for Industrial and Applied Mathematics, Philadelphia, 1995.
Dostupno na: `https://my.siam.org/books/textbooks/fr16_book.pdf` (stranica posjećena: 20. kolovoza 2018.)

[7] C. Sert, *Numerical Methods, Solving Systems of Linear Algebraic Equations*, Middle East Technical University, Mechanical Engineering Department, Ankara, 2006.
Dostupno na `http://users.metu.edu.tr/csert/me310/me310_4_linearSystems.pdf` (stranica posjećena: 20. kolovoza 2018.)

[8] C. Greif, *Numerical Solution of Linear Systems*, Tel Aviv University, 2008. Dostupno na `http://www.cs.tau.ac.il/~dcor/Graphics/adv-slides/Solving.pdf` (stranica posjećena: 20. kolovoza 2018.)

Sažetak

Za rješavanje linearnih sustava jednadžbi postoji nekoliko metoda. Neke od njih su direktne (Gaussove eliminacije, LR faktorizacija) i daju točno rješenje do na pogrešku u zaokruživanju računala, ali nisu praktične za veće sustave jer imaju kubnu složenost. S druge strane su iterativne metode (Jacobijeva, Gauss-Seidelova) koje su brže i efikasne za veće sustave, ali daju aproksimaciju rješenja i u nekim slučajevima sporo konvergiraju prema egzaktnom rješenju. Odabir bolje metode ovisi o danom problemu. U ovom radu su obrađene neke direktne i iterativne metode, te je prezentirana njihova usporedba na primjeru rješavanja diferencijalne jednadžbe. Svi prezentirani algoritmi zapisani su u programskom jeziku Python 3.6.

Summary

There are several methods for solving linear systems of equations. Some of them are direct (Gaussian Elimination, LR Decomposition) and provide the exact solution neglecting the roundoff error caused by the computer, but they are not practical for solving large systems because their complexity is cubic. On the other side, there are iterative methods (Jacobi, Gauss-Seidel) which are faster and more efficient to use in large systems, but they give approximate solutions and their convergence to the exact solution can be slow. Choosing a better method depends on a given problem. This thesis deals with some of direct and iterative methods, and shows their comparison on the example of solving differential equations. All presented algorithms are written in a programming language Python 3.6.

Životopis

Karla Mikec rođena je 2.1.1995. u Čakovcu. Živi u Donjem Kraljevcu gdje je polazila osnovnu školu od 2001. do 2009. godine. Srednjoškolsko obrazovanje nastavila je u Gimnaziji Josipa Slavenskog u Čakovcu gdje je upisala smjer jezične gimnazije. Godine 2013. završila je srednju školu te položila državnu maturu. Iste je godine upisala preddiplomski sveučilišni studij Matematika; smjer: nastavnički na Prirodoslovno-matematičkom fakultetu u Zagrebu. Preddiplomski studij je završila 2016. godine kada je upisala diplomski sveučilišni studij Matematika; smjer nastavnički na istom fakultetu. Na drugoj godini diplomskog sveučilišnog studija imala je priliku odraditi nekoliko sati kao zamjena za profesoricu matematike u V. gimnaziji u Zagrebu.