Poglavje 2

Sistemi linearnih enačb

Najpogostejši problem, na katerega naletimo pri numeričnem računanju, je reševanje sistema linearnih enačb. Tak sistem lahko dobimo direktno iz matematične formulacije fizikalnega problema, ali pa je vmesna stopnja pri reševanju kakšnega zapletenejšega problema, npr. aproksimacije funkcije s polinomom, reševanju navadnih ali parcialnih diferencialnih enačb z diferenčno metodo, reševanju sistemov nelinearnih enačb, ...

V tem poglavju bomo najprej predstavili osnovne pojme v zvezi s sistemi linearnih enačb in njihovo rešljivostjo, v nadaljevanju pa si bomo ogledali nekaj metod za numerično reševanje sistemov linearnih enačb. Začeli bomo z algoritmoma za reševanje spodnje trikotnega in zgornje trikotnega sistema, nato pa si bomo podrobneje ogledali Gaussovo eliminacijsko metodo in njeno varianto, LU razcep. Da bi se izognili nekaterim težavam pri reševanju, bomo opisali tehniko delnega pivotiranja, na kratko pa si bomo ogledali tudi, kako lahko ocenimo natančnost izračunane rešitve. Na koncu se bomo spoznali še z dvema iteracijskima metodama, z Jacobijevo in z Gauss-Seidlovo metodo, ki imata prednost pred Gaussovo eliminacijasko metodo predvsem pri reševanju zelo velikih linearnih sistemov, ki jih najpogosteje srečujemo pri računanju rešitev parcialnih diferencialnih enačb.

2.1 Predstavitev problema

Sistem m linearnih enačb z n neznankami lahko zapišemo kot

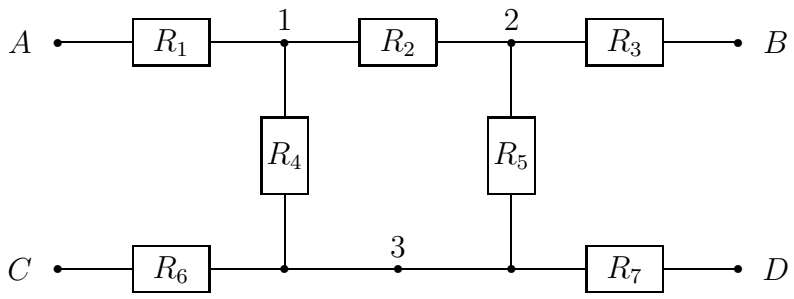
$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & = & b_m. \end{array} \tag{2.1}$$

Koeficienti sistema a_{ij} $(i = 1, \ldots, m; j = 1, \ldots, n)$ in *desne strani* b_i $(i = 1, \ldots, m)$ so dana števila. Poiskati moramo (če je to mogoče) števila x_j $(j = 1, \ldots, n)$ tako, da bodo enačbe (2.1) vse hkrati izpolnjene. Sistem enačb (2.1) navadno zapišemo v matrični obliki

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{2.2}$$

kjer je $A = [a_{ij}]_{m\times n}$ matrika koeficientov (osnovna matrika sistema), $\boldsymbol{b} = [b_i]_m$ vektor desnih strani in $\boldsymbol{x} = [x_j]_n$ vektor neznank.

Primer 2.1.1. Kot primer uporabe sistema linearnih enačb pri reševanju konkretnega problema vzemimo električno vezje na sliki 2.1. Pri danih vrednostih uporov $R_1, R_2, \ldots, R_7$ in danih napetostih U_A, U_B, U_C in U_D v točkah A, B, C in D nas zanimajo napetosti U_1, U_2 in U_3 v točkah 1, 2 in 3.



Slika 2.1: Električno vezje

Za reševanje uporabimo Kirchhoffov zakon[1]: vsota električnih tokov, ki

[1]Gustav Robert Kirchhoff (1824 Königsberg – 1887 Berlin), nemški fizik, ki se je ukvarjal predvsem elektrotehniko in analizo spektrov. Svoja zakona o tokovih v električnih vezjih je objavil leta 1945.

tečejo v vsako vozlišče električnega vezja je enaka 0. To zapišemo simbolično $\sum I = 0$, kjer je $I = \Delta U/R$.

Tako dobimo za vsoto tokov v točkah 1, 2 in 3 enačbe

$$\begin{aligned} \frac{U_1 - U_A}{R_1} + \frac{U_1 - U_2}{R_2} + \frac{U_1 - U_3}{R_4} &= 0 \\ \frac{U_2 - U_1}{R_2} + \frac{U_2 - U_B}{R_3} + \frac{U_2 - U_3}{R_5} &= 0 \\ \frac{U_3 - U_C}{R_6} + \frac{U_3 - U_1}{R_4} + \frac{U_3 - U_2}{R_5} + \frac{U_3 - U_D}{R_7} &= 0. \end{aligned}$$

Ko te enačbe preuredimo, dobimo linearni sistem

$$A\boldsymbol{U} = \boldsymbol{R},$$

kjer je

$$A = \begin{bmatrix} R_8 & -R_1R_4 & -R_1R_2 \\ -R_3R_5 & R_9 & -R_2R_3 \\ -R_5R_6R_7 & -R_4R_6R_7 & R_{10}, \end{bmatrix}$$

(tu smo pisali $R_8 = R_1R_2 + R_1R_4 + R_2R_4$, $R_9 = R_2R_3 + R_2R_5 + R_3R_5$ in $R_{10} = R_4R_5R_6 + R_4R_5R_7 + R_4R_6R_7 + R_5R_6R_7$),

$$\boldsymbol{R} = \begin{bmatrix} R_2R_4U_A \\ R_2R_5U_B \\ R_4R_5R_7U_C + R_4R_5R_6U_D \end{bmatrix} \quad \text{in} \quad \boldsymbol{U} = \begin{bmatrix} U_1 \\ U_2 \\ U_3. \end{bmatrix}$$

■

Preden se lotimo reševanja sistema (2.2), povzemimo znani rezultat iz linearne algebre o rešljivosti sistemov linearnih enačb:

Izrek 2.1.1. *Za linearni sistem $A\boldsymbol{x} = \boldsymbol{b}$ veljajo naslednje trditve:*

1. *Sistem ima rešitev natanko tedaj, ko je rang razširjene matrike $[A \,|\, \boldsymbol{b}]$ enak rangu osnovne matrike:*

$$\text{rang}[A \,|\, \boldsymbol{b}] = \text{rang}A.$$

2. *Sistem ima največ eno rešitev natanko tedaj, ko ima ustrezen homogen sistem $A\boldsymbol{x} = \boldsymbol{0}$ samo trivialno rešitev $\boldsymbol{x} = \boldsymbol{0}$.*

3. *Vsak homogen sistem* $A\boldsymbol{x} = \mathbf{0}$*, ki ima manj enačb kakor neznank, ima netrivialne rešitve.*

4. *Če ima sistem* (2.2) *rešitev za poljuben vektor* $\boldsymbol{b}$*, potem število enačb ni večje od števila neznank* $m \leq n$*.*

Običajno nas zanimajo taki sistemi $A\boldsymbol{x} = \boldsymbol{b}$, ki imajo pri vsaki desni strani b natanko eno rešitev. Zaradi Izreka 2.1.1 se lahko omejimo na sisteme enačb s kvadratno matriko, to je sisteme, kjer je število enačb enako številu neznank.

Izrek 2.1.2. *Naj bo število enačb* m *v sistemu* $A\boldsymbol{x} = \boldsymbol{b}$ *enako številu neznank* n*. Potem so enakovredne naslednje trditve:*

1. *Homogen sistem* $A\boldsymbol{x} = \mathbf{0}$ *ima le trivialno rešitev* $\boldsymbol{x} = \mathbf{0}$*.*

2. *Sistem* (2.2) *ima natanko eno rešitev za poljuben vektor* $\boldsymbol{b}$*.*

3. *Če sistem* (2.2) *za nek* $\boldsymbol{b}$ *ima rešitev, je ta ena sama.*

4. *Vrstice (stolpci) matrike* A *so linearno neodvisni vektorji.*

5. $\det A \neq 0$*.*

6. *Obstaja inverzna matrika* A^{-1}*, da je* $AA^{-1} = A^{-1}A = I$*.*

7. $\boldsymbol{x} = A^{-1}\boldsymbol{b}$*.*

Omejili se bomo na reševanje nesingularnih sistemov, to so sistemi z nesingularno ($\det A \neq 0$) matriko. V tem primeru lahko rešitev zapišemo s pomočjo determinant po Cramerjevemu[2] pravilu, vendar s praktičnega stališča ta rešitev ni zanimiva, saj računanje vrednosti ene determinante zahteva približno ravno toliko računskih operacij, kot reševanje celotnega sistema linearnih enačb.

Pripomniti moramo tudi, da je pogoj $\det A \neq 0$ pri numeričnem reševanju zelo težko preverljiv, saj se vrednost determinante zelo spremeni, če vse enačbe sistema (2.1) pomnožimo z istim faktorjem α. Za determinanto matrike, pomnožene s skalarjem α velja

$$\det(\alpha A) = \alpha^n \det A.$$

[2]Gabriel Cramer (1704 Ženeva – 1752 Bagnols-sur-Cèze), švicarski matematik, ukvarjal se je predvsem z geometrijo in zgodovino matematike. Svoje pravilo za rešitev sistema linearnih enačb je objavil leta 1750.

Za primer vzemimo sistem reda 30 (kar je dokaj majhen sistem za današnje standarde) in vse enačbe pomnožimo z 1/10. Tako je

$$\det\left(\frac{1}{10}A\right) = 10^{-30}\det A,$$

kar pomeni, da deljenje vseh enačb sistema z 10 zmanjša determinanto sistema za faktor 10^{-30}, kar je praktično težko razlikovati od 0. Zato test rešljivosti z determinanto uporabljamo bolj v teoretičnih razmišljanjih.

2.2 Trikotni sistemi

Preden se lotimo reševanja splošnega sistema (2.1), si oglejmo, kako lahko rešimo sistem s trikotno matriko. Matrika L je *spodnja trikotna*, če za njene elemente l_{ij} velja, da je $l_{ij} = 0$ za $i < j$. Podobno je U zgornja trikotna matrika, če za njene elemente u_{ij} velja, da je $u_{ij} = 0$ za $i > j$. Preprosto povedano: zgornja trikotna matrika ima ima elemente, različne od 0, le na glavni diagonali in nad njo, spodnja trikotna matrika pa ima elemente, različne od 0, le na glavni diagonali in pod njo. Tako spodnjo trikotno matriko L reda 4 lahko zapišemo kot

$$L = \begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix}.$$

Poglejmo si najprej, kako rešimo sistem linearnih enačb reda 3 s spodnjo trikotno matriko. Najprej sistem enačb zapišimo:

$$\begin{aligned} l_{11}x_1 + 0 \quad\quad + 0 \quad\quad &= b_1 \\ l_{21}x_1 + l_{22}x_2 + 0 \quad\quad &= b_2 \\ l_{31}x_1 + l_{32}x_2 + l_{33}x_1 &= b_3 \end{aligned} . \tag{2.3}$$

Ker je determinanta tega sistema enaka produktu diagonalnih elementov, $\det L = l_{11}l_{22}l_{33}$, je sistem enolično rešljiv natanko tedaj, ko so diagonalni elementi vsi različni od 0.

Iz prve enačbe izračunamo $x_1 = b_1/l_{11}$, to vrednost vstavimo v drugo enačbo in jo rešimo

$$x_2 = \frac{b_2 - l_{21}x_1}{l_{22}},$$

nato pa vstavimo dobljeni vrednosti za x_1 in za x_2 v tretjo enačbo, iz katere lahko izračunano še vrednost neznanke x_3:

$$x_3 = \frac{b_3 - l_{31}x_1 - l_{32}x_2}{l_{33}}.$$

Če so koeficienti $l_{11},\ l_{22}$ in l_{33} različni od nič, smo rešili sistem (2.3).

Pri reševanju splošnega spodnje trikotnega sistema $L\boldsymbol{x} = \boldsymbol{b}$ reda n moramo i-to enačbo rešiti glede na neznanko x_i:

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} l_{ij}x_j}{l_{ii}}.$$

To moramo ponoviti po vrsti za $i = 1, \ldots, n$. Ker desno stran b_i potrebujemo le pri reševanju i-te enačbe, kasneje pa nič več, lahko na njeno mesto v pomnilniku zapišemo vrednost rešitve x_i. Tako dobimo algoritem, ki ga imenujmo *direktno vstavljanje*. Zapišimo ga kot zaporedje operacij nad elementi ustreznih matrik:

Algoritem 2.2.1 (Direktno vstavljanje)**.** Naj bo L spodnja trikotna nesingularna matrika reda n in $\boldsymbol{b}$ n-dimenzionalni vektor (stolpec). Naslednji algoritem na mesto, kjer je bil vektor $\boldsymbol{b}$, zapiše rešitev sistema $L\boldsymbol{x} = \boldsymbol{b}$.

$b(1) = b(1)/L(1,1)$
for $i = 2 : n$
    for $j = 1 : i-1$
        $b(i) = b(i) - L(i,j) * b(j)$
    end
    $b(i) = b(i)/L(i,i)$
end

Isti algoritem lahko zapišemo v bolj kompaktni obliki, če uporabimo MATLABov zapis s podmatrikami:

$b(1) = b(1)/L(1,1)$
for $i = 2 : n$
    $b(i) = (b(i) - L(i, 1 : i-1) * b(1 : i-1))/L(i,i)$
end

Preštejmo aritmetične operacije, potrebne za izvedbo tega algoritma. V notranji zanki, ki je v kompaktnem zapisu skrita v skalarnem produktu $L(i, 1 : i-1) * b(1 : i-1)$, imamo eno množenje. Notranja zanka se izvede $i-1$ krat, torej imamo $i-1$ množenje. V zunanji zanki, ki se izvede $n-1$ krat, imamo poleg notranje zanke še eno deljenje, ki ga kar prištejemo množenjem, kar nam da skupaj

$$1 + \sum_{i=2}^{n}(1 + \sum_{j=1}^{i-1} 1) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \approx \frac{n^2}{2}$$

množenj in deljenj. Podobno je število seštevanj in odštevanj. Zato pravimo, da je časovna zahtevnost opisanega algoritma enaka $n^2/2$.

Ustreznemu algoritmu za reševanje sistema $U\boldsymbol{x} = \boldsymbol{b}$ z zgornje trikotno matriko U pravimo *obratno vstavljanje*. Enačbe rešujemo od spodaj (od zadnje proti prvi), neznanko x_i pa izračunamo po formuli

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} u_{ij}x_j}{u_{ii}}$$

in spet lahko vektor desnih strani b kar prepišemo z vektorjem rešitev x. Tako dobimo:

Algoritem 2.2.2 (Obratno vstavljanje)**.** Naj bo U zgornja trikotna nesingularna matrika reda n in $\boldsymbol{b}$ n-dimenzionalni vektor. Naslednji algoritem nadomesti vektor $\boldsymbol{b}$ z rešitvijo sistema $U\boldsymbol{x} = \boldsymbol{b}$.

$b(n) = b(n)/U(n, n)$
for $i = n-1 : -1 : 1$
    for $j = i+1 : n$
        $b(i) = b(i) - U(i, j) * b(j)$
    end
    $b(i) = b(i)/U(i, i)$
end

Tudi ta algoritem zapišimo še v kompaktni obliki s podmatrikami:

$b(n) = b(n)/U(n, n)$
for $i = n-1 : -1 : 1$
    $b(i) = (b(i) - U(i, i+1 : n) * b(i+1 : n))/U(i, i)$
end

Podobno kot algoritem z direktnim vstavljanjem ima tudi algoritem z obratnim vstavljanjem časovno zahtevnost $n^2/2$.

2.3 Gaussova eliminacijska metoda

Kot smo ravnokar videli, je reševanje trikotnih sistemov enostavno, zato to poizkusimo izkoristiti tudi pri sistemih, ki nimajo trikotne oblike. Ideja Gaussove[3] eliminacijske metode je splošen kvadratni sistem $A\boldsymbol{x} = \boldsymbol{b}$ preoblikovati v enakovreden trikotni sistem. To dosežemo s primernimi linearnimi kombinacijami enačb. Da bi se izognili težavam, bomo predpostavili, da ima matrika A vse vodilne poddeterminante $\det A(1:k, 1:k)$; $k = 1, \ldots, n$ različne od 0. Kako ravnamo, če ta pogoj ni izpolnjen, pa si bomo ogledali v razdelku 2.5.

Primer 2.3.1. Vzemimo kot primer naslednji sistem reda 2:

$$\begin{array}{rcrcr} 2x_1 & + & 3x_2 & = & 8 \\ 4x_1 & + & 7x_2 & = & 18. \end{array}$$

Če prvo enačbo, pomnoženo z 2, odštejemo od druge, dobimo enakovreden zgornje trikotni sistem

$$\begin{array}{rcl} 2x_1 & +3x_2 & = 8 \\ & x_2 & = 2, \end{array}$$

od koder lahko z obratnim vstavljanjem izračunamo rešitev

$$x_2 = 2 \quad \text{in} \quad x_1 = \frac{8 - 3 \cdot 2}{2} = 1.$$

■

Če postopek, ki smo ga uporabili za reševanje primera 2.3.1 posplošimo na sistem n enačb, ga lahko formalno zapišemo kot:

Algoritem 2.3.1 (Gaussova eliminacija)**.** Naj bo dana kvadratna matrika A reda n z lastnostjo, da so vse njene glavne podmatrike $A(1:k, 1:k)$; $k = 1, \ldots, n$ nesingularne, in naj bo $\boldsymbol{b}$ n-dimenzionalni vektor. Naslednji

[3] Johann Carl Friedrich Gauss (1777 Brunswick – 1855 Göttingen), nemški matematik, fizik in astronom, eden najpomembnejših matematikov. Ukvarjal se je skoraj z vsemi področji matematike. Svojo metodo za reševanje sistemov linearnih enačb je razvil na osnovi stare kitajske metode, opisane v tekstu *Devet poglavij iz umetnosti matematike* iz obdobja okoli 200 pr. n. št., da bi lahko izračunal tirnico asteroida Pallas.

algoritem preoblikuje sistem linearnih enačb $A\boldsymbol{x} = \boldsymbol{b}$ v enakovreden zgornje trikotni sistem $U\boldsymbol{x} = \boldsymbol{c}$. Ko je algoritem končan, so elementi zgornje trikotne matrike U zapisani v zgornjem trikotniku matrike A, preoblikovani vektor desnih strani $\boldsymbol{c}$ pa se nahaja na mestu vektorja $\boldsymbol{b}$.

for $k = 1 : n - 1$
  for $i = k + 1 : n$
    $M(i, k) = A(i, k)/A(k, k)$
    for $j = k + 1 : n$
      $A(i, j) = A(i, j) - M(i, k) * A(k, j)$
    end
    $b(i) = b(i) - M(i, k) * b(k)$
  end
end

Da bi izračunali rešitve prvotnega sistema $A\boldsymbol{x} = \boldsymbol{b}$, moramo le še z obratnim vstavljanjem rešiti zgornje trikotni sistem $U\boldsymbol{x} = \boldsymbol{c}$.

Preštejmo še število množenj/deljenj, ki je potrebno za preoblikovanje sistema z Gaussovo eliminacijo

$$\sum_{k=1}^{n-1} \left(\sum_{i=k+1}^{n} \left(2 + \sum_{j=k+1}^{n} 1 \right) \right) = \frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6} \approx \frac{n^3}{3},$$

torej ima Gaussova eliminacija časovno zahtevnost $n^3/3$.

Primer 2.3.2. Sistem enačb

$$\begin{array}{rcrcrcr} 3x_1 & + & 4x_2 & + & x_3 & = & 6 \\ 5x_1 & + & 5x_2 & + & x_3 & = & 6 \\ -2x_1 & + & 2x_2 & + & 4x_3 & = & 10 \end{array}$$

rešimo z Gaussovo eliminacijsko metodo.

Začnimo z razširjeno matriko sistema

$$\left[\begin{array}{rrr|r} 3 & 4 & 1 & 6 \\ 5 & 5 & 1 & 6 \\ -2 & 2 & 4 & 10 \end{array}\right].$$

Po prvem koraku Gaussove eliminacije ($k = 1$) dobimo

$$\left[\begin{array}{rrr|r} 3 & 4 & 1 & 6 \\ 0 & -1.667 & -0.6667 & -4 \\ 0 & 4.667 & 4.667 & 14 \end{array}\right]$$

in po drugem koraku ($k = 2$)

$$\left[\begin{array}{ccc|c} 3 & 4 & 1 & 6 \\ 0 & -1.667 & -0.6667 & -4 \\ 0 & 0 & 2.8 & 2.8 \end{array}\right],$$

nato pa z obratnim vstavljanjem izračunamo rešitev

$$x_1 = -1, \qquad x_2 = 2, \qquad x_3 = 1.$$

■

2.4 LU Razcep

V algoritmu 2.3.1 smo zgornji trikotnik (vključno z glavno diagonalo) matrike A prepisali z zgornje trikotno matriko U, ki smo jo dobili kot rezultat Gaussove transformacije matrike A. Definirajmo še spodnje trikotno matriko $L = [m_{ik}]$, katere (i, k)-ti element naj bo faktor m_{ik}, s katerim smo množili k-to vrstico, preden smo jo odšteli od i-te, na glavni diagonali pa naj bodo enojke:

$$L = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ m_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ m_{n1} & \dots & m_{n,n-1} & 1 \end{bmatrix}. \tag{2.4}$$

Za matrike U, L in originalno matriko A velja naslednji rezultat:

Izrek 2.4.1. *Naj bo dana nesingularna matrika A, matrika U naj bo določena z algoritmom* 2.3.1*, matrika L pa s formulo* (2.4)*. Potem velja*

$$LU = A. \tag{2.5}$$

Dokaz: Naj bo $M_k\,(k = 1, \dots, n-1)$ matrika, ki jo dobimo, če v n razsežni enotski matriki v k-tem stolpcu pod glavno diagonalo namesto 0 pišemo $-m_{ik}$; $i = k+1, \dots, n$. Tako je

$$M_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -m_{21} & 1 & \cdots & 0 \\ -m_{31} & 0 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ -m_{n1} & 0 & \cdots & 1 \end{bmatrix}, \quad M_2 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ 0 & -m_{32} & \cdots & 0 \\ & \vdots & \ddots & \vdots \\ 0 & -m_{n2} & \cdots & 1 \end{bmatrix}.$$

Izračunajmo

$$A_1 = M_1 A = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -m_{21} & 1 & \cdots & 0 \\ -m_{31} & 0 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ -m_{n1} & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ a_{31} & a_{32} & \cdots & a_{3n} \\ & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a'_{22} & \cdots & a'_{2n} \\ 0 & a'_{32} & \cdots & a'_{3n} \\ & \vdots & \ddots & \vdots \\ 0 & a'_{n2} & \cdots & a'_{nn} \end{bmatrix}$$

Tako je A_1 matrika, ki dobimo iz A po prvem koraku Gaussove eliminacije. Podobno izračunamo po vrsti še

$$A_2 = M_2 A_1, \ \ldots, \ A_{n-1} = M_{n-1} A_{n-2}$$

in ugotovimo, da je $A_{n-1} = U$ zgornje trikotna matrika, ki jo dobimo pri Gaussovi eliminaciji. Tako imamo

$$M_{n-1} \cdots M_2 M_1 A = U.$$

Da bi dokazali trditev izreka, moramo to enačbo najprej pomnožiti z leve zaporedoma z $M_{n-1}^{-1}, \ldots, M_2^{-1}$ in M_1^{-1}, da dobimo

$$A = M_1^{-1} M_2^{-1} \cdots M_{n-1}^{-1} U,$$

nato pa izračunati produkt $M_1^{-1} M_2^{-1} \cdots M_{n-1}^{-1}$. Inverzne matrike M_k^{-1} dobimo iz matrik M_k tako, da zamenjamo predznake faktorjev m_{ij}, torej

$$M_1^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ m_{21} & 1 & \cdots & 0 \\ m_{31} & 0 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ m_{n1} & 0 & \cdots & 1 \end{bmatrix}, \ M_2^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ 0 & m_{32} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & m_{n2} & \cdots & 1 \end{bmatrix}, \qquad (2.6)$$

in podobno za ostale, kar lahko enostavno preverimo z direktnim računom (glej problem 3).

Izračunati moramo še produkt $M_1^{-1}M_2^{-1}\dots M_{n-1}^{-1}$. Najprej zmnožimo $M_1^{-1}M_2^{-1}$:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ m_{21} & 1 & \cdots & 0 \\ m_{31} & 0 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ m_{n1} & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ 0 & m_{32} & \cdots & 0 \\ & \vdots & \ddots & \vdots \\ 0 & m_{n2} & \cdots & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ m_{21} & 1 & \cdots & 0 \\ m_{31} & m_{32} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & \cdots & 1 \end{bmatrix}.$$

Tako nadaljujemo, dokler na koncu ne dobimo spodnje trikotne matrike $L = M_1^{-1}M_2^{-1}\dots M_{n-1}^{-1}$, tako da velja enačba (2.5). Tako smo dokazali veljavnost izreka. QED

Zapišimo še algoritem, ki nam izračuna LU razcep matrike A:

Algoritem 2.4.1 (LU razcep)**.** Naj bo A kvadratna matrika reda n z lastnostjo, da so vse njene podmatrike $A(1\!:\!k, 1\!:\!k);\ k = 1, \dots, n$ nesingularne. Naslednji algoritem izračuna razcep $A = LU$, kjer je L spodnje trikotna matrika z enojkami na glavni diagonali, U pa zgornje trikotna matrika. Ko je algoritem končan, so elementi matrike U zapisani v zgornjem trikotniku matrike A, elementi matrike L, razen enojk na diagonali, pa v spodnjem trikotniku matrike A.

for $k = 1 : n-1$
  for $i = k+1 : n$
    $A(i,k) = A(i,k)/A(k,k)$
    for $j = k+1 : n$
      $A(i,j) = A(i,j) - A(i,k) * A(k,j)$
    end
  end
end

Preštejmo še množenja in deljenja, potrebna za LU razcep matrike z algoritmom 2.4.1:

$$\sum_{k=1}^{n-1}\left[\sum_{i=k+1}^{n}\left(1+\sum_{j=k+1}^{n}1\right)\right] = \sum_{k=1}^{n-1}\left[n-k+(n-k)^2\right] = \frac{n^3}{3} - \frac{n}{3} \approx \frac{n^3}{3}.$$

Časovna zahtevnost algoritma za LU razcep je torej $n^3/3$, prav tako kot časovna zahtevnost algoritma Gaussove eliminacije.

Kako si lahko z LU razcepom matrike A pomagamo pri reševanju sistema enačb (2.2)? Če smo matriko A razcepili v produkt LU, lahko sistem zapišemo kot

$$LU\boldsymbol{x} = \boldsymbol{b}$$

ki ga rešimo v dveh korakih. Najprej z direktnim vstavljanjem rešimo spodnje trikotni sistem

$$L\boldsymbol{y} = \boldsymbol{b},$$

s pomožnim vektorjem $\boldsymbol{y} = U\boldsymbol{x}$, nato pa še z obratnim vstavljanjem zgornje trikotni sistem

$$U\boldsymbol{x} = \boldsymbol{y}.$$

Izračunajmo potrebno število množenj/deljenj za rešitev sistema (2.2) po opisanem postopku. Za LU razcep potrebujemo $n^3/3 - n/3$ operacij, za direktno vstavljanje $n^2/2 - n/2$ (tukaj smo že upoštevali, da imamo na glavni diagonali enice) in $n^2/2 + n/2$ za obratno vstavljanje. Skupaj je to $n^3/3 + n^2 - n/3$ množenj/deljenj, kar je skoraj enako kot pri sami Gaussovi eliminaciji.

Glavnino operacij pri reševanju linearnega sistema predstavlja LU razcep, saj je število operacij za reševanje obeh trikotnih sistemov za velikostni red manjše. Tako se glavna prednost LU razcepa pokaže, kadar moramo zaporedoma rešiti več sistemov linearnih enačb (2.2) z isto matriko sistema A in različnimi desnimi stranmi. V tem primeru je potreben le en LU razcep, za vsak vektor desnih strani pa moramo rešiti po dva trikotna sistema, za kar pa je potrebno znatno manj (približno n^2) množenj/deljenj.

Primer 2.4.1. Za primer izračunajmo rešitev sistema linearnih enačb

$$\begin{aligned} 3x_1 + 2x_2 + 5x_3 + x_4 &= 1 \\ 6x_1 + 6x_2 + 15x_3 + 3x_4 &= -6 \\ -3x_1 + 4x_2 + 13x_3 + x_4 &= -17 \\ -6x_1 + 6x_2 + 15x_3 + 5x_4 &= -52. \end{aligned} \tag{2.7}$$

Najprej izračunajmo LU razcep matrike sistema

$$A = \begin{bmatrix} 3 & 2 & 5 & 1 \\ 6 & 6 & 15 & 3 \\ -3 & 4 & 13 & 1 \\ -6 & 6 & 15 & 5 \end{bmatrix}$$

s pomočjo algoritma 2.4.1. Spremljajmo, kaj se dogaja z matriko v posameznih korakih. Po prvem koraku ($k = 1$) dobimo namesto elementov matrike A

$$\begin{bmatrix} 3 & 2 & 5 & 1 \\ 2 & 2 & 5 & 1 \\ -1 & 6 & 18 & 2 \\ -2 & 10 & 25 & 7 \end{bmatrix},$$

po drugem koraku ($k = 2$) imamo

$$\begin{bmatrix} 3 & 2 & 5 & 1 \\ 2 & 2 & 5 & 1 \\ -1 & 3 & 3 & -1 \\ -2 & 5 & 0 & 2 \end{bmatrix},$$

po tretjem koraku ($k = 3$) pa se matrika ne spremeni, ker že ima 0 na mestu (4,3). Tako sta matriki L in U enaki

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ -1 & 3 & 1 & 0 \\ -2 & 5 & 0 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 3 & 2 & 5 & 1 \\ 0 & 2 & 5 & 1 \\ 0 & 0 & 3 & -1 \\ 0 & 0 & 0 & 2 \end{bmatrix}.$$

Rešiti moramo še oba trikotna sistema. Najprej $L\boldsymbol{y} = \boldsymbol{b}$ z direktnim vstavljanjem:

$$\begin{aligned} y_1 &= 1 \\ y_2 &= -6 - 2y_1 = -8 \\ y_3 &= -17 + y_1 - 3y_2 = 8 \\ y_4 &= -52 + 2y_1 - 5y_2 + 0y_3 = -10, \end{aligned}$$

nato pa še zgornje trikotni sistem $U\boldsymbol{x} = \boldsymbol{y}$ z obratnim vstavljanjem:

$$\begin{aligned} x_4 &= \frac{-10}{2} = -5 \\ x_3 &= \frac{8 + x_4}{3} = 1 \\ x_2 &= \frac{-8 - x_4 - 5x_3}{2} = -4 \\ x_1 &= \frac{1 - x_4 - 5x_3 - 2x_2}{3} = 3 \end{aligned}$$

Rešitev sistema (2.7) je torej

$$\boldsymbol{x} = [3, -4, 1, -5]^T.$$

■

2.5 Pivotiranje

V algoritmih za Gaussovo eliminacijo in za LU razcep (algoritma 2.3.1 in 2.4.1) smo predpostavili, da ima matrika A vse vodilne poddeterminante $\det A(1:k, 1:k)$; $k = 1, \ldots, n$ različne od 0. S tem smo preprečili možnost, da bi bil kateri od diagonalni elementov, s katerim delimo, in ki ga običajno imenujemo *pivot*, lahko enak 0. Poglejmo si, kako se lahko izognemo delitvi z nič, če ta pogoj ni izpolnjen.

Denimo, da je na k-tem koraku LU razcepa pivot $a_{kk} = 0$. Če je matrika A nesingularna, mora biti v k-tem stolpcu pod glavno diagonalo vsaj en element različen od 0. Denimo, da je $a_{ik} \neq 0$; $k + 1 \leq i \leq n$. V tem primeru lahko med seboj zamenjamo k-to in i-to vrstico, saj zamenjava dveh vrstic v matriki sistema ustreza zamenjavi dveh enačb, kar seveda rešitve sistema ne spremeni. Zgodi pa se, da tudi v primeru, ko so vsi pivotni elementi različni od 0, nismo zadovoljni z rešitvijo. Oglejmo si naslednji primer:

Primer 2.5.1. Rešimo sistem linearnih enačb

$$\begin{aligned} 6x_1 &+ 2x_2 &+ 2x_3 &= -2 \\ 2x_1 &+ \tfrac{2}{3}x_2 &+ \tfrac{1}{3}x_3 &= 1 \\ x_1 &+ 2x_2 &- x_3 &= 0, \end{aligned} \tag{2.8}$$

katerega točna rešitev je

$$x_1 = 2.6, \quad x_2 = -3.8 \quad \text{in} \quad x_3 = -5.0. \tag{2.9}$$

Računali bomo na štiri decimalna mesta z zaokrožanjem. Najprej zapišimo matriko sistema

$$\begin{bmatrix} 6.000 & 2.000 & 2.000 \\ 2.000 & 0.6667 & 0.3333 \\ 1.000 & 2.000 & -1.000 \end{bmatrix},$$

in izračunajmo njen LU razcep. Po prvem koraku je

$$\begin{bmatrix} 6.000 & 2.000 & 2.000 \\ 0.3333 & 0.0001000 & -0.3333 \\ 0.1667 & 1.667 & -1.333 \end{bmatrix}$$

in po drugem koraku

$$\begin{bmatrix} 6.000 & 2.000 & 2.000 \\ 0.3333 & 0.0001000 & -0.3333 \\ 0.1667 & 16670 & 5555 \end{bmatrix}.$$

Rešimo najprej spodnje trikotni sistem

$$\begin{aligned} y_1 &= -2.000 \\ y_2 &= 1.000 - 0.3333y_1 = 1.667 \\ y_3 &= 0 - 0.1667y_1 - 16670y_2 = -27790, \end{aligned}$$

nato pa še zgornje trikotnega

$$\begin{aligned} x_3 &= \frac{-27790}{5555} = -5.003 \\ x_2 &= \frac{1.667 + 0.3333x_3}{0.0001000} = 0.000 \\ x_1 &= \frac{-2.000 - 2.000x_3 - 2.000x_2}{6.000} = 1.335, \end{aligned}$$

kar se precej razlikuje od točne rešitve (2.9).

Vir težav v tem primeru je pivot na drugem koraku, ki bi moral biti enak 0, pa zaradi zaokrožitvene napake v prejšnjem koraku ni. Če bi pred drugim korakom zamenjali drugo in tretjo vrstico matrike, bi bil rezultat LU razcepa

$$\begin{bmatrix} 6.000 & 2.000 & 2.000 \\ 0.1667 & 1.667 & -1.333 \\ 0.3333 & 0.00005999 & -0.3332 \end{bmatrix}.$$

Sedaj dobimo iz spodnje trikotnega sistema

$$\begin{aligned} y_1 &= -2.000 \\ y_2 &= 0 - 0.1667y_1 = 0.3334 \quad \text{(Zamenjali smo 2. in 3. enačbo!)} \\ y_3 &= 1.000 - 0.3333y_1 - 0.00006000y_2 = 1.667, \end{aligned}$$

nato pa še iz zgornje trikotnega sistema

$$\begin{aligned} x_3 &= \frac{1.667}{-0.3332} = -5.003 \\ x_2 &= \frac{0.3334 + 1.333x_3}{1.667} = -3.801 \\ x_1 &= \frac{-2.000 - 2.000x_3 - 2.000x_2}{6.000} = 2.602, \end{aligned}$$

kar se mnogo bolje ujema s točno rešitvijo (2.9). ■

Problemom, kot v prejšnjem primeru, se večinoma lahko izognemo z zamenjavo vrstnega reda enačb, čemur pravimo *delno pivotiranje*. Na k-tem koraku Gaussove eliminacije (ali LU razcepa) izberemo za pivot po absolutni vredosti največji element v k-tem stolpcu od k-te do zadnje vrstice

$$c = \max_{k \le i \le n} |a_{ik}|.$$

Če je $|a_{kk}| < c$, potem zamenjamo k-to vrstico z eno od naslednjih, da dobimo po zamenjavi $|a_{kk}| = c$. Na ta način izberemo za pivot tisti element, ki je kar se da daleč od 0. Z delnim pivotiranjem dosežemo, da so vsi faktorji m_{ik}, torej vsi elementi matrike L, omejeni z 1

$$|l_{ij}| \le 1,$$

kar se pokaže kot ugodno tudi zaradi zaokrožitvenih napak.

Seveda pa izrek 2.5 za pivotiranje ne velja več v isti obliki. Pokazati se da, da je

$$PA = LU; \quad L\boldsymbol{y} = P\boldsymbol{b}; \quad U\boldsymbol{x} = \boldsymbol{y},$$

kjer je matrika P (pravimo ji *permutacijska matrika*) dobljena iz enotske matrike z istimi zamenjavami vrstic, kot jih zahteva delno pivotiranje za matriko A.

Dopolnimo algoritem 2.4.1 za izračun LU razcepa z delnim pivotiranjem:

Algoritem 2.5.1 (LU z delnim pivotiranjem)**.** Naj bo A kvadratna, nesingularna matrika reda n. Naslednji algoritem izračuna razcep $PA = LU$, kjer je L spodnje trikotna matrika z enojkami na glavni diagonali, U pa spodnje trikotna matrika. Ko je algoritem končan, so elementi matrike U zapisani v zgornjem trikotniku matrike A, elementi matrike L, od katerih nobeden ni po absolutni vrednosti večji od 1 pa v spodnjem trikotniku matrike A. V celoštevilskem vektorju $\boldsymbol{p}$ so zapisane podrobnosti o zamenjavah vrstic.

for $k = 1 : n - 1$
    $pivot = abs(A(k, k))$
    $p(k) = k$
    for $i = k + 1 : n$
        if $abs(A(i, k)) > pivot$
            $pivot = abs(A(i, k))$
            $p(k) = i$

```
        end
    end                          %Element A_ik je največji
    if pivot = 0
        error('Matrika je singularna')
    end
    temp = A(k,:)                %Zamenjamo k-to in p(k)-to vrstico
    A(k,:) = A(p(k),:)
    A(p(k),:) = temp
    for i = k+1 : n
        A(i,k) = A(i,k)/A(k,k)
        for j = k+1 : n
            A(i,j) = A(i,j) - A(i,k) * A(k,j)
        end
    end
end
```

2.6 Metoda Choleskega

Splošni sistem linearnih enačb $A\boldsymbol{x} = \boldsymbol{b}$ najučinkoviteje rešimo tako, da faktoriziramo matriko $A = LU$, kjer je L spodnje trikotna in U zgornje trikotna matrika, nato pa rešimo dobljena trikotna sistema. Kadar pa je matrika A simetrična, je ugodno, če je tudi faktorizacija simetrična, na primer $A = R^T R$, kjer je R zgornje trikotna matrika. Žal na ta način ne moremo faktorizirati vsake simetrične matrike.

Poglejmo, kakšna mora biti matrika A, da bo imela simetrično faktorizacijo. Naj bo R nesingularna zgornje trikotna matrika, in $\boldsymbol{x}$ poljuben neničelen vektor. Potem je tudi $A = R^T R$ nesingularna in $\boldsymbol{y} = R\boldsymbol{x} \neq 0$. Tako je

$$\boldsymbol{x}^T A\boldsymbol{x} = \boldsymbol{x}^T R^T R\boldsymbol{x} = (R\boldsymbol{x})^T (R\boldsymbol{x}) = \boldsymbol{y}^T \boldsymbol{y} = \sum y_i^2 > 0.$$

Zato definirajmo:

Definicija 2.6.1. *Matrika A, za katero je kvadratna forma $\boldsymbol{x}^T A\boldsymbol{x} > 0$ za poljuben vektor $\boldsymbol{x} \neq 0$, se imenuje* pozitivno definitna.

Navedimo dve preprosti lastnosti pozitivno definitnih matrik, ki jih bomo kasneje s pridom uporabili:

1. Pozitivno definitna matrika je nesingularna.

   Dokaz. Če je matrika A singularna, obstaja neničelen vektor x, da je $Ax = 0$. Tako je tudi $x^T Ax = 0$ in matrika A ni pozitivno definitna. QED

2. Če je matrika $A = [a_{ij}]_{i,j=1}^n$ pozitivno definitna, potem je matrika $A_* = [a_{ij}]_{i,j=2}^n$ tudi pozitivno definitna in $a_{11} > 0$.

   Dokaz. Matriko A zapišimo v bločni obliki

   $$A = \begin{bmatrix} a_{11} & a \\ b & A_* \end{bmatrix},$$

   kjer je $a = [a_{12}, a_{13}, \ldots, a_{1n}]$ in $b = [a_{21}, a_{31}, \ldots, a_{n1}]^T$. Naj bo $y \in \mathbb{R}^{n-1}$ poljuben neničelen vektor in $x^T = [0, y^T]$. Potem je

   $$0 < x^T Ax = [0, y^T] \begin{bmatrix} a_{11} & a \\ b & A_* \end{bmatrix} \begin{bmatrix} 0 \\ y \end{bmatrix} = y^T A_* y$$

   in A_* je pozitivno definitna. Če pa vzamemo $x = [1, 0, \ldots, 0]^T$, je

   $$0 < x^T Ax = [1, 0] \begin{bmatrix} a_{11} & a \\ b & A_* \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = a_{11}.$$

   QED

Pozitivno definitne simetrične matrike nastopajo pri reševanju najrazličnejših problemov dovolj pogosto, da se splača njihovo simetrijo izkoristiti pri reševanju sistemov linearnih enačb.

Algoritem za izračun elementov zgornje trikotne matrike R izpeljemo najlaže, če enačbo $A = R^T R$ zapišemo v bločni obliki tako, da prvi stolpec in prvo vrstico matrik A in R zapišemo posebej:

$$\begin{bmatrix} \alpha & a^T \\ a & A_* \end{bmatrix} = \begin{bmatrix} \rho & 0 \\ r & R_*^T \end{bmatrix} \cdot \begin{bmatrix} \rho & r^T \\ 0 & R_* \end{bmatrix}. \tag{2.10}$$

Ko to matrično enačbo zapišemo po blokih, dobimo tri enačbe

$$\begin{aligned} \alpha &= \rho^2 \\ a^T &= \rho r^T \\ A_* &= R_*^T R_* + r r^T, \end{aligned}$$

od koder je

$$\begin{aligned} \rho &= \sqrt{\alpha} \\ r^T &= a^T/\rho \\ R_*^T R_* &= A_* - rr^T \end{aligned} \tag{2.11}$$

S pomočjo prvih dveh enačb lahko izračunamo prvo vrstico matrike R, zadnjo enačbo pa si poglejmo podrobneje. Matrika

$$\hat{A}_* = A_* - rr^T = A_* - \frac{aa^T}{\alpha}$$

je simetrična, ker je

$$\hat{A}_*^T = (A_* - rr^T)^T = A_*^T - (rr^T)^T = A_* - rr^T = \hat{A}_*.$$

Trditev 2.6.1. *Matrika $\hat{A}_*$ je pozitivno definitna.*

Dokaz. Pokazati moramo, da je za vsak neničelen vektor y

$$y^T \hat{A}_* y = y^T A_* y - (a^T y)^2/\alpha > 0.$$

Ker je matrika A pozitivno definitna, je za poljubno število η

$$0 < [\eta \quad y^T] \begin{bmatrix} \alpha & a^T \\ a & A_* \end{bmatrix} \begin{bmatrix} \eta \\ y \end{bmatrix} = \alpha\eta^2 + 2\eta a^T y + y^T A_* y.$$

Če si izberemo $\eta = -a^T y/\alpha$, dobimo oceno

$$0 < \alpha\eta^2 + 2\eta a^T y + y^T A_* y = y^T A_* y - (a^T y)^2/\alpha,$$

kar že pomeni, da je matrika $\hat{A}_*$ pozitivno definitna. QED

Dimenzija kvadratne matrike $\hat{A}_*$ je za 1 manjša od dimenzije prvotne matrike A, zato je matrika R_* v enačbi (2.11) faktor v razcepu Choleskega za matriko $\hat{A}_*$, ki ga lahko izračunamo rekurzivno.

Algoritem 2.6.1 (Razcep Choleskega)**.** [4] Naj bo A simetrična pozitivno definitna matrika reda n. Naslednji algoritem izračuna zgornjo trikotno matriko R, da je $A = R^T R$. Ob koncu so elementi matrike R zapisani v zgornjem trikotniku matrike A.

[4] Andre-Louis Cholesky (1875 Montguyon – 1918), francoski oficir in geodet. Metodo, danes poimenovano po njem, je razvil, da bi reševal normalne sisteme enačb, ki nastanejo pri uporabi metode najmanjših kvadratov. Padel je tri mesece pred koncem I. svetovne vojne.

for $k = 1 : n$
  $A(k,k) = \mathbf{sqrt}(A(k,k))$
  for $i = k+1 : n$
    $A(k,i) = A(k,i)/A(k,k)$
  end
  for $i = k+1 : n$
    for $j = i : n$
      $A(i,j) = A(i,j) - A(k,j) * A(k,i)$
    end
  end
end

Tudi ta algoritem zapišimo še v kompaktni obliki s podmatrikami:

for $k = 1 : n$
  $A(k,k) = \mathbf{sqrt}(A(k,k))$
  $A(k,k+1:n) = A(k,k+1:n)/A(k,k)$
  for $i = k+1 : n$
    $A(i,i:n) = A(i,i:n) - A(k,i:n) * A(k,i)$
  end
end

Preštejmo množenja in deljenja, ki so potrebne za razcep Choleskega z algoritmom 2.6.1:

$$\sum_{k=1}^{n}\left(\sum_{i=k+1}^{n} 1 + \sum_{i=k+1}^{n}(\sum_{j=i}^{n} 1)\right) = \sum_{k=1}^{n}\left(n-k+\sum_{i=k+1}^{n}(n-i+1)\right)$$
$$= \sum_{k=1}^{n}\left(n-k+\frac{(n-k)(n-k+1)}{2}\right) = \frac{(n-1)n(n+4)}{6} \approx \frac{n^3}{6},$$

poleg tega pa še n kvadratnih korenov. Število množenj in deljenj, potrebnih za razcep simetrične, pozitivno definitne matrike z algoritmom 2.6.1 je torej približno pol manjše kot za LU razcep z algoritmom 2.4.1.

Primer 2.6.1. Izračunajmo rešitev sistema linearnih enačb $A\boldsymbol{x} = \boldsymbol{b}$, kjer sta

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}, \quad \boldsymbol{b} = [1,0,0,1]^T.$$

Pri razcepu Choleskega dobimo

$$R = \begin{bmatrix} 1.414213562 & -0.707106781 & 0 & 0 \\ 0 & 1.224744871 & -0.816496581 & 0 \\ 0 & 0 & 1.154700538 & -0.866025404 \\ 0 & 0 & 0 & 1.118033989 \end{bmatrix},$$

po direktnem vstavljanju $R^T\boldsymbol{y} = \boldsymbol{b}$ in obratnem vstavljanju $R\boldsymbol{x} = \boldsymbol{y}$ izračunamo rešitev

$$\boldsymbol{x} = [1, 1, 1, 1]^T.$$

■

2.7 Napaka in ostanek

Zaradi zaokrožitvenih napak je vsaka izračunana rešitev sistema linearnih enačb le približek k pravi rešitvi. V tem razdelku si bomo ogledali, kako lahko ocenimo napako izračunane rešitve.

Naj bo $\boldsymbol{x}$ točna in $\hat{\boldsymbol{x}}$ izračunana rešitev linearnega sistema $A\boldsymbol{x} = \boldsymbol{b}$. *Napaka* $\boldsymbol{e}$ izračunane rešitve je torej enaka

$$\boldsymbol{e} = \hat{\boldsymbol{x}} - \boldsymbol{x}. \tag{2.12}$$

Ker je napaka vektor, potrebujemo primerno mero za ocenitev njegove velikosti. Tako kot 'velikost' realnega števila določa njegova absolutna vrednost, določa 'velikost' vektorja njegova *norma*.

V matematiki je norma vektorja (normo vektorja $\boldsymbol{x}$ zapišemo s simbolom $||\boldsymbol{x}||$) definirana kot funkcija z definicijskim območjem v vektorskem prostoru in zalogo vrednosti v realnih številih, ki zadošča naslednjim zahtevam[5]:

1. $||\boldsymbol{x}|| \geq 0$ za vsak vektor $\boldsymbol{x}$ in $||\boldsymbol{x}|| = 0$ le za $\boldsymbol{x} = \boldsymbol{0}$.

2. $||\alpha\boldsymbol{x}|| = |\alpha| \cdot ||\boldsymbol{x}||$ za vsak skalar α in vsak vektor $\boldsymbol{x}$.

3. $||\boldsymbol{x} + \boldsymbol{y}|| \leq ||\boldsymbol{x}|| + ||\boldsymbol{y}||$ za vsak par vektorjev $\boldsymbol{x}$ in $\boldsymbol{y}$ istih dimenzij.

[5]Tem zahtevam ustrezajo tudi absolutne vrednosti realnih in kompleksnih števil ter dolžine vektorjev

Prva lastnost zahteva, da je norma neničelnih vektorjev pozitivna. Druga lastnost zahteva, da imata vektorja $\boldsymbol{x}$ in $-\boldsymbol{x}$ isto normo in da se norma vektorja pomnoži z absolutno vrednostjo skalarja, če pomnožimo vektor s skalarjem. Tretja lastnost norme je poznana pod imenom *trikotniška neenakost*, ker zahteva, da mora biti dolžina vsake stranice trikotnika manjša od vsote dolžin ostalih dveh stranic.

Iz linearne algebre je znana *evklidska norma*

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

pri numeričnem računanju pa največkrat uporabljamo tako imenovano *maksimum* normo

$$||\boldsymbol{x}||_\infty = \max_{1\leq i\leq n} |x_i|.$$

Lahko se je prepričati, da tudi za maksimum normo veljajo vse tri lastnosti norme.

Primer 2.7.1. Za vektor $\boldsymbol{a} = [1, 2, 3]^{\mathrm{T}}$ je

$$||\boldsymbol{a}||_2 = \sqrt{1+4+9} = \sqrt{14} \approx 3.74166 \quad \text{in} \quad ||\boldsymbol{a}||_\infty = 3.$$

■

Podobno lahko 'velikost' matrik merimo z *matrično* normo, ki mora zadoščati naslednjim lastnostim:

1. Za vsako kvadratno matriko A je $||A|| \geq 0$ in $||A|| = 0$ le za ničelno matriko $A = 0$.
2. $||\alpha A|| = |\alpha| \cdot ||A||$ za vsako kvadratno matriko A in vsak skalar α.
3. $||A + B|| \leq ||A|| + ||B||$ za vsak par kvadratnih matrik A in B istih dimenzij.
4. $||AB|| \leq ||A|| \cdot ||B||$ za vsak par kvadratnih matrik A in B istih dimenzij.

Iz vsake vektorske norme lahko dobimo ustrezno matrično normo s predpisom

$$||A|| = \max_{\boldsymbol{x}\neq 0} \frac{||A\boldsymbol{x}||}{||\boldsymbol{x}||}, \tag{2.13}$$

tako da velja ocena

$$||A\boldsymbol{x}|| \leq ||A|| \cdot ||\boldsymbol{x}||.$$

Na ta način je matrična maksimum norma definirana s pomočjo vektorske maksimum norme [3]

$$||A||_\infty = \max_{\boldsymbol{x} \neq 0} \frac{||A\boldsymbol{x}||_\infty}{||\boldsymbol{x}||_\infty},$$

in je enaka

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

medtem ko je matrično normo, definirano s pomočjo vektorske evklidske norme

$$||A||_2 = \max_{\boldsymbol{x} \neq 0} \frac{||A\boldsymbol{x}||_2}{||\boldsymbol{x}||_2}$$

običajno precej teže izračunati.

Primer 2.7.2. Za matriko

$$A = \begin{bmatrix} 4 & -6 & 2 \\ 0 & 4 & 1 \\ 1 & 2 & 3 \end{bmatrix}$$

je $||A||_\infty = \max\{12, 5, 6\} = 12$, medtem ko je $||A||_2 \approx 8.1659$ ■

Vrnimo se k obravnavi ocen za napako (2.12) rešitve sistema linearnih enačb. Če izračunano rešitev vstavimo v sistem (2.2), dobimo *ostanek* $\boldsymbol{r} = A\hat{\boldsymbol{x}} - \boldsymbol{b}$, za katerega velja

$$\boldsymbol{r} = A\hat{\boldsymbol{x}} - A\boldsymbol{x} = A(\hat{\boldsymbol{x}} - \boldsymbol{x}) = A\boldsymbol{e}, \tag{2.14}$$

oziroma $\boldsymbol{e} = A^{-1}\boldsymbol{r}$, od koder hitro dobimo oceno za napako izračunane rešitve

$$||\boldsymbol{e}|| = ||A^{-1}\boldsymbol{r}|| \leq ||A^{-1}|| \cdot ||\boldsymbol{r}||. \tag{2.15}$$

Podobno oceno za relativno napako $||\boldsymbol{e}||/||\boldsymbol{x}||$ dobimo tako, da enačbo (2.15) delimo z $||\boldsymbol{x}||$ in upoštevamo, da je $||A|| \cdot ||\boldsymbol{x}|| \geq ||\boldsymbol{b}||$:

$$\frac{||\boldsymbol{e}||}{||\boldsymbol{x}||} \leq ||A^{-1}|| \cdot ||A|| \frac{||\boldsymbol{r}||}{||\boldsymbol{b}||}, \tag{2.16}$$

Ta ocena velja za poljubno vektorsko normo in z njeno pomočjo definirano matrično normo. Povejmo še z besedami: relativna napaka je kvečjemu za faktor $||A^{-1}|| \cdot ||A||$ večja od relativnega ostanka. Število $||A^{-1}|| \cdot ||A||$ je značilno za posamezno matriko. Imenujemo ga *število občutljivosti* (angl. *condition number*) in ga zapišemo

$$\text{cond}(A) = ||A^{-1}|| \cdot ||A||,$$

če pa je matrika A singularna (in A^{-1} ne obstaja), tedaj je $\text{cond}(A) = \infty$.

Zaradi lastnosti matrične norme je $||I|| = ||A^{-1}A|| \leq ||A^{-1}|| \cdot ||A||$, po drugi strani pa je, za matrične norme, ki so definirane s pomočjo vektorske norme, $||I|| = \max ||A\boldsymbol{x}||/||\boldsymbol{x}|| = 1$, zato je število občutljivosti $\text{cond}(A) \geq 1$ za vsako matriko A. Matrikam, katerih število občutljivosti je majhno (to pomeni: ne dosti večje od 1), pravimo *dobro pogojene* matrike. Nasprotno so *slabo pogojene* tiste matrike, katerih število občutljivosti je veliko.

Ocena (2.16) pomeni, da je pri dobro pogojenih matrikah ostanek dobra ocena za napako v rešitvi, pri slabo pogojenih matrikah pa iz majhnega ostanka ne moremo sklepati, da je tudi napaka majhna.

Da bi lahko izračunali število občutljivosti dane matrike A, bi morali poznati njeno inverzno matriko A^{-1}, kar pa je navadno še težja naloga kot rešiti sistem linearnih enačb.

Število občutljivosti je obratno sorazmerno oddaljenosti matrike od najbljižje singularne matrike:

Izrek 2.7.1. *[9] Naj bo $A \in \mathbb{R}^{n\times n}$ nesingularna matrika. Za vsako singularno matriko $B \in \mathbb{R}^{n\times n}$ velja*

$$\frac{1}{\text{cond}(A)} \leq \frac{||A - B||}{||A||}.$$

Dokaz: Zaradi definicije matrične norme (2.13) je

$$||A^{-1}|| = \max_{\boldsymbol{x}\neq 0} \frac{||A^{-1}\boldsymbol{x}||}{||\boldsymbol{x}||} \geq \frac{||A^{-1}\boldsymbol{x}||}{||\boldsymbol{x}||}$$

za vsak vektor $\boldsymbol{x} \in \mathbb{R}^n$, torej tudi

$$||A^{-1}|| \geq \frac{||\boldsymbol{y}||}{||A\boldsymbol{y}||}$$

za vsak vektor $\boldsymbol{y} \in \mathbb{R}^n$. Od tod sklepamo, da tudi ocena

$$\frac{1}{\text{cond}(A)} \leq \frac{1}{||A||} \frac{||A\boldsymbol{y}||}{||\boldsymbol{y}||}$$

velja za vsak vektor $\boldsymbol{y}$. Izberimo neničelen vektor $\boldsymbol{y}$ tako, da bo $B\boldsymbol{y} = 0$. Tak vektor zagotovo obstaja, ker je matrika B singularna. Tako velja

$$\frac{1}{\text{cond}(A)} \leq \frac{||(A-B)\boldsymbol{y}||}{||A|| \cdot ||\boldsymbol{y}||} \leq \frac{||A-B|| \cdot ||\boldsymbol{y}||}{||A|| \cdot ||\boldsymbol{y}||} = \frac{||A-B||}{||A||}.$$

QED

S pomočjo izreka 2.7.1 lahko ugotovimo, kako daleč je matrika A od najbližje singularne matrike: če je $\text{cond}(A)$ veliko število, je A skoraj singularna. Zato pri reševanju sistema linearnih enačb z zelo občutljivo matriko lahko dobimo velike napake.

Tudi brez poznavanja števila občutljivosti lahko ugotovimo, ali je sistem slabo ali dobro pogojen, ob tem pa še izboljšamo točnost izračunane rešitve s tehniko *iterativnega izboljševanja*.

Vzemimo, da imamo približno rešitev $\hat{\boldsymbol{x}}$ sistema (2.2), in da smo izračunali ostanek $\boldsymbol{r}^1 = A\hat{\boldsymbol{x}} - \boldsymbol{b}$. Vemo, da v tem primeru napaka izračunane rešitve zadošča enačbi (2.14), katere matrika je ista kot matrika prvotnega sistema. Ker smo prvotni sistem že rešili, torej poznamo njegov LU razcep, lahko tudi do rešitve sistema (2.14) pridemo z rešitvijo dveh trikotnih sistemov, kar pomeni le n^2 množenj. Ko tako poznamo napako $\boldsymbol{e}$, lahko 'popravimo' izračunano rešitev, saj je

$$\hat{\boldsymbol{x}}^1 = \hat{\boldsymbol{x}} - \boldsymbol{e}$$

boljši približek za rešitev kot $\hat{\boldsymbol{x}}$. Na ta način lahko nadaljujemo, da dobimo zaporedje rešitev $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{x}}^1$, $\hat{\boldsymbol{x}}^2, \ldots$, kot tudi zaporedje ustreznih ostankov $\hat{\boldsymbol{r}}$, $\hat{\boldsymbol{r}}^1$, $\hat{\boldsymbol{r}}^2, \ldots$. Iz števila pravilnih mest približkov $\hat{\boldsymbol{x}}^i$, kot tudi iz konvergence zaporedja ostankov lahko sklepamo na natančnost teh približkov rešitve. Kadar je število $\text{cond}(A)$ veliko, zaporedje rešitev počasneje ali sploh ne konvergira proti pravi rešitvi. Tako lahko sklepamo, da je sistem zelo slabo pogojen, kadar zaporedje ostankov le počasi ali pa sploh ne konvergira proti 0.

POZOR! Posebej velja opozoriti, da moramo pri iterativnem izboljševanju rešitve ostanke $\boldsymbol{r}^i$ računati čim bolj natančno, navadno v dvojni natančnosti.

Algoritem 2.7.1 (Iterativno izboljšanje)**.** Naj bo dan linearen sistem $A\boldsymbol{x} = \boldsymbol{b}$, LU razcep matrike A in približna rešitev $\hat{\boldsymbol{x}}$. Naslednji algoritem izračuna boljši približek za rešitev:

Izračunaj ostanek $\boldsymbol{r} = A\hat{\boldsymbol{x}} - \boldsymbol{b}$ v dvojni natančnosti
Reši sistem $A\boldsymbol{e} = \boldsymbol{r}$ z uporabo algoritmov z direktnim in obratnim vstavljanjem
$\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}} - \boldsymbol{e}$
Če je $||\boldsymbol{e}||/||\hat{\boldsymbol{x}}||$ dovolj majhno število, naj bo $\hat{\boldsymbol{x}}$ rešitev, sicer ponovi postopek

Če smo že izračunali rešitev $\hat{\boldsymbol{x}}$ s pomočjo LU razcepa matrike A, se skoraj vedno izplača naknadno iterativno izboljšanje rešitve, saj zanjo za vsak korak iteracije potrebujemo le rešitev dveh trikotnih sistemov. Iz konvergence iterativnega izboljšanja pa lahko sklepamo na občutljivost sistema.

S številom občutljivosti $\mathrm{cond}(A)$ lahko tudi ocenimo, kako je rešitev linearnega sistema odvisna od spremembe desne strani $\boldsymbol{b}$.

Izrek 2.7.2. *Naj bo matrika A nesingularna. Za točne rešitve sistemov $A\boldsymbol{x} = \boldsymbol{b}$ in $A\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}$ velja ocena*

$$\frac{||\boldsymbol{x} - \hat{\boldsymbol{x}}||}{||\boldsymbol{x}||} \leq \mathrm{cond}(A)\frac{||\boldsymbol{b} - \hat{\boldsymbol{b}}||}{||\boldsymbol{b}||}. \tag{2.17}$$

Dokaz: Odštejmo $A\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}$ od $A\boldsymbol{x} = \boldsymbol{b}$

$$A(\boldsymbol{x} - \hat{\boldsymbol{x}}) = \boldsymbol{b} - \hat{\boldsymbol{b}},$$

oziroma

$$\boldsymbol{x} - \hat{\boldsymbol{x}} = A^{-1}(\boldsymbol{b} - \hat{\boldsymbol{b}}).$$

Zaradi zveze med vektorsko in matrično normo je

$$||\boldsymbol{x} - \hat{\boldsymbol{x}}|| = ||A^{-1}(\boldsymbol{b} - \hat{\boldsymbol{b}})|| \leq ||A^{-1}||\,||\boldsymbol{b} - \hat{\boldsymbol{b}}||.$$

Delimo obe strani neenačbe z $||\boldsymbol{x}||$

$$\frac{||\boldsymbol{x} - \hat{\boldsymbol{x}}||}{||\boldsymbol{x}||} \leq \frac{||A^{-1}||\,||\boldsymbol{b} - \hat{\boldsymbol{b}}||}{||\boldsymbol{x}||} = \frac{||A||\,||A^{-1}||\,||\boldsymbol{b} - \hat{\boldsymbol{b}}||}{||A||\,||\boldsymbol{x}||}. \tag{2.18}$$

Ker, spet zaradi zveze med vektorsko in matrično normo, velja tudi

$$||\boldsymbol{b}|| = ||A\boldsymbol{x}|| \leq ||A||\,||\boldsymbol{x}||,$$

kar uporabimo na desni strani neenačbe (2.18), in dobimo oceno (2.17). QED

Na tem mestu le omenimo, da lahko s pomočjo LU razcepa izračunamo tudi determinanto kvadratne matrike in inverzno matriko. Iz razcepa $A = LU$ je očitno, da je $\det A = \det L \cdot \det U$, ker pa je $\det L = 1$ (matrika L je trikotna z enojkami na diagonali), je

$$\det A = \prod_{i=1}^{n} u_{ii},$$

saj je tudi U trikotna matrika.

Inverzna matrika A^{-1} je rešitev matrične enačbe $AA^{-1} = I$, ki jo lahko zapišemo po stolpcih kot zaporedje linearnih sistemov z isto matriko A:

$$A\boldsymbol{x}^{(j)} = \boldsymbol{e}^{(j)}; \quad j = 1, \ldots, n,$$

kjer so $\boldsymbol{x}^{(j)}$ stolpci inverzne matrike A^{-1}, $\boldsymbol{e}^{(j)}$ pa ustrezni stolpci enotske matrike I. Opozoriti velja, da inverzno matriko le redkokdaj potrebujemo, saj je praviloma učinkoviteje rešiti ustrezen sistem linearnih enačb. Tako namesto računanja produkta $C = A^{-1}B$ raje rešujemo matrično enačbo $AC = B$.

2.8 Iterativne metode

Pri reševanju nekaterih problemov, predvsem pri reševanju parcialnih diferencialnih enačb, naletimo na linearne sisteme z zelo velikim številom neznank (tudi $n = 10000$ ali več). Običajno ima pri teh sistemih matrika A le nekaj elementov v vsaki vrstici različnih od 0 (taki matriki pravimo *razpršena matrika* (angl. *sparse matrix*) za razliko od *goste matrike* (angl. *dense matrix*), pri kateri je večina elementov različna od 0). Gaussova eliminacijska metoda za velike razpršene sisteme ni primerna, ker zahteva preveč računanja (spomnimo se: za rešitev sistema reda n potrebujemo približno $n^3/3$ množenj in deljenj), poleg tega pa se med računanjem 'napolnijo' tudi tisti elementi, ki so bili v prvotni matriki A enaki 0, kar lahko predstavlja znatne težave pri shranjevanju elementov v pomnilniku računalnika.

Zaradi tega za reševanje zelo velikih razpršenih sistemov raje od direktne metode, kot je Gaussova eliminacija ali LU razcep, uporabljamo iteracijske metode. To so metode, pri katerih tvorimo zaporedje približnih rešitev $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots$, ki pri določenih pogojih konvergira proti rešitvi prvotnega sistema. Da bo iteracijska metoda učinkovita, mora biti izračun novega

približka enostaven, z majhnim številom računskih operacij, zaporedje približkov pa mora dovolj hitro konvergirati proti pravilni rešitvi. Ogledali si bomo dve enostavni iteracijski metodi, Jacobijevo in Gauss-Seidlovo.

Jacobijeva iteracija Jacobijevo metodo[6] najlaže izpeljemo tako, da v linearnem sistemu (2.2) i-to enačbo rešimo glede na spremenljivko x_i. Tako dobimo

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j - \sum_{j=i+1}^{n} a_{ij} x_j \right); \quad i = 1, \ldots, n. \tag{2.19}$$

Izberimo si začetni približek k rešitvi $\boldsymbol{x}^{(0)}$ (zgornji indeksi v oklepaju bodo pomenili zaporedno številko iteracije). Novi, izboljšani približek k rešitvi dobimo tako, da prejšnji približek vstavimo v desno stran enačbe (2.19). Tako pridemo do splošnega koraka Jacobijeve iteracijske sheme

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right); \quad i = 1, \ldots, n, \tag{2.20}$$

ki jo lahko nekoliko poenostavimo, če na desni strani prištejemo in odštejemo $x_i^{(k)}$:

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{n} a_{ij} x_j^{(k)} \right); \quad i = 1, \ldots, n. \tag{2.21}$$

Poglejmo si še, kako Jacobijevo iteracijsko shemo zapišemo v matrični obliki. Najprej matriko A zapišimo v obliki $A = S + D + Z$, kjer je matrika S spodnji trikotnik matrike A brez diagonale, D diagonalna matrika, ki vsebuje diagonalne elemente matrike A, matrika Z pa zgornji trikotnik matrike A brez diagonale. Sistem (2.20) lahko sedaj zapišemo kot

$$D\boldsymbol{x}^{(k+1)} = \boldsymbol{b} - (S + Z)\boldsymbol{x}^{(k)}. \tag{2.22}$$

Ker je matrika D diagonalna, je ta sistem lahko rešljiv, če so le vsi diagonalni koeficienti $a_{ii} \neq 0$. Uporabo Jacobijeve metode si oglejmo na primeru:

[6] Carl Gustav Jacob Jacobi (1804 Potsdam – 1851 Berlin), namški matematik, poznan predvsem po teoriji eliptičnih funkcij in funkcijskih determinantah. Iterativno metodo za reševanje sistemov linearnih enačb je objavil leta 1845 v astronomski reviji.

Primer 2.8.1. Rešujemo sistem enačb

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \tag{2.23}$$

katerega rešitev je $x_1 = x_2 = x_3 = x_4 = 1$.

Vzemimo začetni približek $x_1 = x_2 = x_3 = x_4 = 0$, naslednje približke pa računamo po formuli (2.20). Iz rezultatov, ki so v tabeli 2.1, vidimo, da napaka s številom iteracij pada in da je po 20 iteracijah približek k rešitvi točen na 2 decimalni mesti, po 50 iteracijah točen na 4 decimalna mesta, po 100 iteracijah pa že na več kot 9 decimalnih mest. ■

k	x_1	x_2	x_3	x_4
1	0.50000000	0	0	0.50000000
2	0.50000000	0.25000000	0.25000000	0.50000000
3	0.62500000	0.37500000	0.37500000	0.62500000
4	0.68750000	0.50000000	0.50000000	0.68750000
5	0.75000000	0.59375000	0.59375000	0.75000000
6	0.79687500	0.67187500	0.67187500	0.79687500
⋮				
20	0.99155473	0.98633527	0.98633527	0.99155473
21	0.99316763	0.98894500	0.98894500	0.99316763
22	0.99447250	0.99105632	0.99105632	0.99447250
⋮				
50	0.99998536	0.99997632	0.99997632	0.99998536
51	0.99998816	0.99998084	0.99998084	0.99998816
52	0.99999042	0.99998450	0.99998450	0.99999042
⋮				
100	1.00000000	1.00000000	1.00000000	1.00000000

Tabela 2.1: Približki rešitve enačbe (2.23), izračunani z Jacobijevo iteracijsko metodo

Pogoje za konvergenco Jacobijeve metode je v praksi težko natančno opredeliti. Velja sicer naslednji izrek o konvergenci, ki ga navajamo brez dokaza:

Izrek 2.8.1. *Zaporedje približkov k rešitvi linearnega sistema* (2.2)*, izračunano z Jacobijevo metodo* (2.22) *pri poljubnem začetnem približku* $\boldsymbol{x}^{(0)}$ *konvergira proti točni rešitvi natanko tedaj, ko vse lastne vrednosti* λ *matrike* $M = D^{-1}(S + Z)$ *zadoščajo neenačbi*

$$|\lambda| < 1.$$

V praksi je zelo težko preverjati, ali je pogoj tega izreka izpolnjen. Navadno si pomagamo s kriterijem *diagonalne dominantnosti*:

Definicija 2.8.1. *Matrika* A *je* diagonalno dominantna po vrsticah*, če je*

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

Če je matrika A diagonalno dominantna, potem Jacobijeva metoda konvergira proti točni rešitvi pri vsakem začetnem približku. Navadno je konvergenca hitrejša, čim bolj je sistem diagonalno dominanten.

Gauss-Seidlova iteracija Pri Jacobijevi metodi računamo komponente novega približka $\boldsymbol{x}^{(k+1)}$ iz starega približka $\boldsymbol{x}^{(k)}$. Konvergenco iteracije bi lahko pospešili, če bi v formuli (2.20) na desni strani uporabili tiste komponente novega približka $\boldsymbol{x}^{(k+1)}$, ki jih že poznamo. Tako dobimo Gauss-Seidlovo[7] iteracijsko shemo

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right); \quad i = 1, \ldots, n. \tag{2.24}$$

Če matriko A razdelimo na vsoto matrik $A = S + D + Z$, kot pri Jacobijevi metodi, lahko Gauss-Seidlovo iteracijo zapišemo v matrični obliki kot

$$(S + D)\boldsymbol{x}^{(k+1)} = \boldsymbol{b} - Z\boldsymbol{x}^{(k)}. \tag{2.25}$$

Matrika $S + D$ je spodnje trikotna, zato sistem enačb (2.25) lahko enostavno rešimo z direktnim vstavljanjem.

Na istem primeru si oglejmo še delovanje Gauss-Seidelove metode:

[7]Philipp Ludwig von Seidel (1821 Zweibrücken – 1896 München), nemški matematik in astronom. Iteracijsko metodo, danes znano pod imenom Gauss-Seidel je objavil leta 1874 kot rezultat sodelovanja z Jacobijem. Gauss je podobno metodo opisal že leta 1845.

Primer 2.8.2. Rešujemo linearni sistem (2.23) z Gauss-Seidlovo iteracijsko metodo. Vzemimo začetno rešitev $x_1 = x_2 = x_3 = x_4 = 0$, naslednje približke pa računamo po formuli (2.24). Rezultati so prikazani v tabeli 2.2. Iz rezultatov vidimo, da v tem primeru Gauss-Seidlova iteracija res konvergira hitreje kot Jacobijeva iteracija. ■

k	x_1	x_2	x_3	x_4
1	0.50000000	0.25000000	0.12500000	0.56250000
2	0.62500000	0.37500000	0.46875000	0.73437500
3	0.68750000	0.57812500	0.65625000	0.82812500
4	0.78906250	0.72265625	0.77539062	0.88769531
5	0.86132812	0.81835937	0.85302734	0.92651367
6	0.90917968	0.88110351	0.90380859	0.95190429
⋮				
20	0.99975953	0.99968523	0.99974534	0.99987267
21	0.99984261	0.99979398	0.99983332	0.99991666
22	0.99989699	0.99986515	0.99989091	0.99994545
⋮				
50	0.99999997	0.99999996	0.99999997	0.99999998
51	0.99999998	0.99999997	0.99999998	0.99999999
52	0.99999999	0.99999998	0.99999999	0.99999999

Tabela 2.2: Približki rešitve enačbe (2.23), izračunani z Gauss-Seidlovo iteracijsko metodo

Če je matrika sistema diagonalno dominantna, potem zaporedje približkov, izračunano z Gauss-Seidlovo iteracijsko metodo, konvergira proti točni rešitvi vsaj tako hitro, kot zaporedje, izračunano z Jacobijevo metodo, običajno pa hitreje.

2.9 Povzetek

V splošnem delimo numerične metode za reševanje sistemov linearnih enačb na direktne, pri katerih računamo razcep matrike in pridemo do rešitve s

končnim številom aritmetičnih operacij, in iterativne, pri katerih tvorimo zaporedje približkov, ki pod določenimi pogoji konvergira proti rešitvi.

Od direktnih metod smo spoznali Gaussovo eliminacijsko metodo z njeno varianto, LU razcepom, ki je primernejša za reševanje sistemov, ki imajo pri istih koeficientih različne desne strani in metodo Choleskega za reševanje simetričnih pozitivno definitnih sistemov. Direktne metode se uporabljajo predvsem pri reševanju manjših sistemov in sistemov, pri katerih je večina koeficientov različna od 0. Pri reševanju sistemov linearnih enačb z direktnimi metodami je praviloma priporočljivo, včasih pa celo potrebno, delno pivotiranje.

Včasih lahko pri reševanju linearnih sistemov s pridom izkoristimo posebno obliko matrike koeficientov. Oglejmo si dva značilna primera:

- *Po stolpcih diagonalno dominantne matrike,* to so matrike, pri katerih je
$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ji}| \quad i = 1, \ldots, n,$$
so v praksi pogoste. Zanje je značilno, da pivotiranje ni potrebno.

- *Pasovne matrike* s širino pasu $2p + 1$ so matrike, za katere je
$$a_{ik} = 0 \quad \text{za} \quad |i - k| > p.$$
Take matrike so zelo pogoste pri numeričnem reševanju diferencialnih enačb. Tudi pasovno strukturo, posebej ko je $p \ll n$ lahko s pridom izkoristimo pri reševanju.

Izmed iterativnih metod smo spozanali Jacobijevo in Gauss-Seidlovo metodo. Iterativne metode se uporabljajo v glavnem pri reševanju velikih sistemov z razpršeno, diagonalno dominantno matriko. Take matrike nastopajo predvsem pri numeričnem reševanju parcialnih diferencialnih enačb.

2.10 Problemi

1. Reši problem iz primera 2.1.1 za podatke: $R_1 = 10k\Omega$, $R_2 = 25k\Omega$, $R_3 = 180k\Omega$, $R_4 = 1.5M\Omega$, $R_5 = 37k\Omega$, $R_6 = 1M\Omega$, $R_7 = 0.5M\Omega$, $U_A = 25V$, $U_B = 70V$, $U_C = 250V$ in $U_D = -75V$.

2. Preštej potrebne operacije (množenja in deljenja) za rešitev zgornje trikotnega sistema enačb $U\boldsymbol{x} = \boldsymbol{b}$ z algoritmom 2.2.2.

3. Preveri, da za matrike M_i v dokazu izreka 2.4.1 veljajo enakosti (2.6).

4. Preveri, da v dokazu izreka 2.4.1 res velja $L = M_1^{-1} M_2^{-1} \cdots M_{n-1}^{-1}$.

5. Matrika A je tridiagonalna, če za njene elemente a_{ij} velja $a_{ij} = 0$ kadar je $|i - j| \geq 2$. Algoritem za LU razcep priredi za reševanje sistemov s tridiagonalno matriko. Kolikšno je število operacij, ki je potrebno za reševanje tridiagonalnega sistema reda n?

6. Matrika A je *zgornja Hessenbergova*[8], če je $a_{ij} = 0$ kadarkoli je $i-j \geq 2$.

   (a) Zapiši eno zgornjo Hessenbergovo matriko reda 5.

   (b) Priredi algoritem za LU razcep za reševanje sistema $A\boldsymbol{x} = \boldsymbol{b}$, kjer je A zgornja Hessenbergova matrika.

   (c) preštej aritmetične operacije, potrebne za rešitev sistema linearnih enačb z zgornjo Hessenbergovo matriko.

7. Izračunaj evklidsko ($||.||_2$) in maksimum normo ($||.||_\infty$) naslednjih vektorjev:

   (a) $\boldsymbol{a} = [1, 3, 5, 2, 3, 1]^T$

   (b) $\boldsymbol{a} = [1, -1, 2, -2]^T$

   (c) $\boldsymbol{a} = [2.31, 3.23, 4.87, -1.22, 2.92]^T$

[8]Karl Hessenberg (1904 – 1959), nemški matematik, ukvarjal se je predvsem z numeričnimi metodami za izračun lastnih vrednosti matrike.