Lineáris egyenletrendszerek megoldása

Kormányos Andor

Komplex Rendszerek Fizikája Tanszék

2019. október 14.

M darab egyenlet N változóval, az a_{ij} és b_j értékek ismertek:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N &= b_2 \\ &\vdots \\ a_{M1}x_1 + a_{M2}x_2 + \ldots + a_{MN}x_N &= b_M \end{aligned}$$

Felírhatjuk indexesés mátrix alakban is:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

Cél: az x_i ismeretlenek meghatározása.

Lineáris egyenletrendszer meghatározottsága

$M = N$ esetén

- Jó esély van konkrét megoldás megtalálására, kivéve ha
- van olyan sor, ami más sorok lineárkombinációjaként előáll
  $\longrightarrow$ ekkor a mátrix *sorok szerint degenerált*, vagy
- a mátrix két oszlopa megegyezik[1],
  $\longrightarrow$ ekkor a mátrix *oszlopok szerint degenerált*
- Ha a négyzetes mátrix sorok vagy oszlopok szerint degenerált
  $\Rightarrow det(\mathbf{A}) = 0 \Rightarrow$ ekkor a mátrix *szinguláris*

[1] pontosabban: ha az egyenletrendszer bizonyos változókat csak teljesen azonos lineárkombinációkban tartalmaz

Numerikusan szinguláris mátrixok

A számítógép a valós számokat véges precizitással ábrázolja, emiatt kerekítési hibák adódnak.

Ha két sor nem teljesen azonos, de nagyon hasonlóak ($\epsilon \simeq 10^{-10}$)

- Ekkor a kerekítési hibák miatt a számolás során előfordulhat 0-val osztás $\Rightarrow$ a mátrix *numerikusan szinguláris*
- Ilyen esetekben a 0-val való osztás miatt NaN vagy Infinity lesz az eredmény

$N \gg 1 \Rightarrow$ az egyenletrendszer megoldása sok műveletet igényel

- A kerekítési hibák összeadódnak
- A program lefut, de a végeredmény hibás lesz
- a kapott **x** megoldás visszahelyettesítésével lehet ellenőrizni a hibát

Numerikusan szinguláris mátrixok kezelése

Speciális algoritmus alkalmazása nélkül

- Néhány 10 változós egyenlet még megoldható
- Néhány 100 egyenlethez duplapontosságú aritmetika kell: double típusú változók használata
- 1000 egyenlet fölött már biztosan jelentkeznek a problémák

A feladat gépigénye:

- a memóriaigény N^2-tel skálázik
- a számításigény N^3-bel skálázik

Speciális alakú mátrixok esetében van gyorsabb megoldás.

- tridiagonális (ld. spline interpoláció)
- Vandermonde-típusú (ld. interpolált polinom együtthatói)

Alulhatározott egyenletrendszerek

Ha $M < N$, vagy $M = N$, de az egyenletrendszer degenerált (kevesebb egyenlet, mint ismeretlen): az egyenletrendszer *alulhatározott*

- Egyáltalán nincsen megoldás, vagy
- A megoldás egy egész altér:
  $\mathbf{x}_p$ egy lehetséges megoldás, és ehhez jön még $N - M$ vektor tetszőleges lineárkombinációja

Túlhatározott egyenletrendszerek

Ha $M > N$ (több egyenlet mint ismeretlen) akkor az egyenletrendszer *túlhatározott*

- Általában nincsen megoldás, de
- Van értelme egy lehetséges legjobb megoldásról beszélni.

$$\min_{x_j} \sum_i (b_i - a_{ij} x_j)^2$$

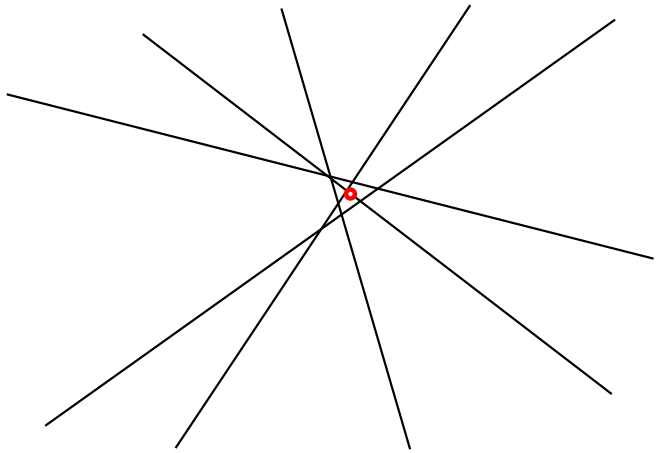
Túlhatározott egyenletrendszerek

Ha $M > N$ (több egyenlet mint ismeretlen) akkor az egyenletrendszer *túlhatározott*

- Általában nincsen megoldás, de
- Van értelme egy lehetséges legjobb megoldásról beszélni.

$$\min_{x_j} \sum_i (b_i - a_{ij} x_j)^2$$

Grafikusan:



Singular value decomposition (SVD)

(Közel) szinguláris, túlhatározott, alulhatározott egyenletrendszer esetén is hasznos lehet az SVD módszer

Házi feladat

Lássuk be, hogy a szinguláris egyenletek legkisebb négyzetek alapján definiált lehető legjobb megoldása előáll a következő egyenletrendszer megoldásaként:

$$\mathbf{A}^T\mathbf{A}\,\mathbf{x} = \mathbf{A}^T\mathbf{b},$$

ahol $\mathbf{A}^T$ az $\mathbf{A}$ mátrix transzponáltja!

- A fenti egyenletrendszert az eredeti egyenletrendszer *normálegyenleteinek* nevezzük.
- Általában ezeket is SVD-vel célszerű megoldani direkt megoldás helyett.

Programcsomagok lineáris egyenletek megoldására

Optimalizált programcsomagok léteznek

- LINPACK, LAPACK
- Párhuzamosított változat: ScaLAPACK
- Intel processzorokra optimalizált: MKL

Támogatják speciális mátrixok optimális tárolását is, pl.:

- Szimmetrikus mátrix, háromszög-mátrix
- Sávmátrixok
- Ritka mátrixok (majdnem minden elem 0)

Gauss–Jordan-elimináció

Feladat: megoldani az alábbi egyenletrendszert:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b},$$

ahol $\mathbf{A}$ négyzetes mátrix.

- Legegyszerűbb általános módszer lineáris egyenletrendszerek megoldására
- Numerikusan legalább annyira stabil, mint más eljárás,
- főleg teljes *pivotolás*[2] esetén

[2] pivot = csuklópont, de a *pivoting* és *pivot element* szavaknak nincsen bevett fordítása

A Gauss–Jordan-elimináció tulajdonságai

Nagy egyenletrenszerek esetén fontos szempontok:

Memóriaigény:

- tárolni kell a mátrixot, és a megoldásvektort: $N^2 + N$
- vagy invertálás esetén a mátrixot és az inverzét: $2N^2$
- a mátrix inverzét elvileg lehet tárolni a bemeneti mátrix helyén

Számítási igény:

- szükséges műveletek száma $O(N^3)$
- A leggyorsabb módszereknél akár háromszor lassabb

Viszont egyszerű, sok mindent meg lehet érteni rajta keresztül

A megoldási módszer elemi lépései

A következő műveletek nem változtatják meg egy lineáris egyenletrendszer megoldását:

- *két sor felcserélése*
  ez nyilvánvaló, hiszen az egyenletek sorrendje teljesen tetszőleges, feltéve persze, hogy a jobboldal megfelelő sorait is megcseréljük
- *más sorok lineárkombinációjának hozzáadása bármely sorhoz*
  ebbe belefér az is, hogy egy sort számmal szorzunk, ha az a szám nem nulla; természetesen mindkét oldalon el kell végezni
- *a változók felcserélése*
  ez lényegileg nem változtat az egyenleteken, ha emlékszünk, hogy a végén a változókat megfelelő permutáció szerint vissza kell cserélgetni; ez az $\mathbf{A}$ mátrix oszlopainak felcserélését jelenti.

Kiindulás: kibővített mátrix

A megoldandó egyenletrendszer:

$$\begin{pmatrix} 2 & 6 & 2 \\ 1 & 1 & -1 \\ 3 & 9 & 5 \end{pmatrix} \cdot \mathbf{x} = \begin{pmatrix} 18 \\ 1 \\ 35 \end{pmatrix}$$

Felírjuk a mátrixot és a jobb oldalt a következő alakban:

$$\left(\begin{array}{ccc|c} 2 & 6 & 2 & 18 \\ 1 & 1 & -1 & 1 \\ 3 & 9 & 5 & 35 \end{array}\right)$$

Ha az inverz mátrixot keressük, akkor pedig:

$$\left(\begin{array}{ccc|ccc} 2 & 6 & 2 & 1 & 0 & 0 \\ 1 & 1 & -1 & 0 & 1 & 0 \\ 3 & 9 & 5 & 0 & 0 & 1 \end{array}\right)$$

- A gyakorlatban a jobb oldalt külön tömbben érdemes tárolni
- Implementáláskor ne másoljuk egybe a két mátrixot!
- A jobb oldalon tulajdonképpen akárhány oszlop állhat, amennyiben egyszerre több egyenletet akarunk megoldani.

Az elimináció menete

Cél: a korábban ismertetett elemi műveletek segítségével

- a baloldalt egységmátrix alakra hozzuk úgy, hogy
- közben a műveleteket a jobb oldalon is elvégezzük
- jobboldalt előáll a mátrix inverze (egyenletrendszer megoldása)

Elindulunk a főátló első elemétől.

1. Ha az elem nem 1, akkor az egész sort elosztjuk a főátlóbeli elemmel, hogy a főátlóban 1 legyen

$$\left(\begin{array}{ccc|c} \mathbf{2} & 6 & 2 & 18 \\ 1 & 1 & -1 & 1 \\ 3 & 9 & 5 & 35 \end{array}\right) \longrightarrow \left(\begin{array}{ccc|c} \mathbf{1} & \mathbf{3} & \mathbf{1} & \mathbf{9} \\ 1 & 1 & -1 & 1 \\ 3 & 9 & 5 & 35 \end{array}\right)$$

2. Minden alatta levő sorból kivonjuk az első sor valahányszorosát úgy, hogy a főátló alatt végig 0 legyen

$$\left(\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 1 & 1 & -1 & 1 \\ 3 & 9 & 5 & 35 \end{array}\right) \longrightarrow \left(\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ \mathbf{0} & -2 & -2 & -8 \\ \mathbf{0} & 0 & 2 & 8 \end{array}\right)$$

3. Ha főátlóbeli elem alatt mindent kinulláztunk, akkor folytatjuk a főátló következő elemével:

$$\left(\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 0 & \mathbf{-2} & -2 & -8 \\ 0 & 0 & 2 & 8 \end{array}\right) \longrightarrow \left(\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & 4 \\ 0 & 0 & 2 & 8 \end{array}\right)$$

4. Majd kivonjuk a sor valahányszorosát az összes többiből úgy, hogy a főátlót kivéve mindenütt 0 legyen:

$$\left(\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 2 & 8 \end{array}\right) \longrightarrow \left(\begin{array}{ccc|c} 1 & \mathbf{0} & -2 & -3 \\ 0 & 1 & 1 & 4 \\ 0 & \mathbf{0} & 2 & 8 \end{array}\right)$$

5. Az eljárást tovább folytatjuk a főátló elemeire

$$\left(\begin{array}{ccc|c} 1 & 0 & -2 & -3 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & \mathbf{2} & 8 \end{array}\right) \longrightarrow \left(\begin{array}{ccc|c} 1 & 0 & -2 & -3 \\ 0 & 1 & 1 & 4 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{4} \end{array}\right)$$

$$\longrightarrow \left(\begin{array}{ccc|c} 1 & 0 & \mathbf{0} & 5 \\ 0 & 1 & \mathbf{0} & 0 \\ 0 & 0 & 1 & 4 \end{array}\right)$$

6. Az eljárás akkor ér végét, ha a baloldali részmátrix az egységmátrix alakját veszi fel. Ekkor jobboldalon vagy a megoldást kapjuk, vagy pedig a mátrix inverzét, attól függően miből indultunk ki.

$$\left(\begin{array}{ccc|c} \mathbf{1} & 0 & 0 & 5 \\ 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 4 \end{array}\right)$$

Zérus elem a főátlóban

Probléma:

$$\left(\begin{array}{ccc|c|c} 1 & 0 & 5 & & \\ 0 & 1 & 6 & & \\ 0 & 0 & \mathbf{0} & \cdots & \cdots \\ & & \vdots & & \end{array}\right)$$

Ilyenkor az adott sorral nem tudjuk az alatta és fölötte levő elemeket eliminálni.

Zérus elem a főátlóban

Probléma:

$$\left(\begin{array}{ccc|c|c} 1 & 0 & 5 & & \\ 0 & 1 & 6 & & \\ 0 & 0 & \mathbf{0} & \cdots & \cdots \\ & & \vdots & & \end{array}\right)$$

Ilyenkor az adott sorral nem tudjuk az alatta és fölötte levő elemeket eliminálni.

Megoldás: sorcsere!
Keressünk a főátló aktuális is eleme alatti valamelyik sorban (de ugyanabban az oszlopban) olyan elemet, ami nem nulla (ún. *pivot-elem*), cseréljük meg a két sort (a jobboldalt is!), majd folytassuk az eliminációt $\rightarrow$ az egyenletek sorrendje tetszőleges !

$\Rightarrow$ *részleges pivotolás*.

Nagyon kis elem a főátlóban

Probléma: ha nagyon kis elemet használnánk a további eliminációs lépésekben, akkor nagy szorzótényezők és emiatt nagyon nagy számok jelennének meg a mátrixban. Ez numerikus instabilitási problémákhoz vezethet.

Megoldás: válasszuk az adott oszlop főátló alatti abszolút értékben legnagyobb elemét, a megfelelő sorokat cseréljük meg, és folytassuk így az eliminációt.

Az egyenletek átskálázása

A sorokat tetszőleges számmal szorozva bármelyik aktuális oszlopbeli, főátló alatti elem lehet maximális. Hogyan válasszuk ki azt az elemet, amellyel a pivotálást végezzük ?

Az egyenletek átskálázása

A sorokat tetszőleges számmal szorozva bármelyik aktuális oszlopbeli, főátló alatti elem lehet maximális. Hogyan válasszuk ki azt az elemet, amellyel a pivotálást végezzük ?

Megoldás: Úgy keressük a legnagyobb elemet, hogy a sorokat az eredeti mátrix sorainak legnagyobb elemével normáljuk.

- Vagy a mátrix sorait a legelején leosztjuk minden sor abszolút értékben maximális elemével,
- Vagy csak eltároljuk a legnagyobb elemeket és a döntésnél használjuk őket, de nem normáljuk explicit a sorokat

$\Rightarrow$*implicit pivotálás*

4. probléma: Az oszlop főátló alatti része csupa 0

$$\left(\begin{array}{cccc|c} 1 & 0 & 5 & & \\ 0 & 1 & 6 & & \\ 0 & 0 & \mathbf{0} & \cdots & \cdots \\ & & \mathbf{0} & & \\ & & \mathbf{0} & & \\ & & \vdots & & \\ & & \mathbf{0} & & \end{array}\right)$$

Ez azt jelenti, hogy a mátrix szinguláris, nincs megoldás.

További lehetőség: oszlopcsere

Eddig csak az sorok cseréjéről volt szó.
Általánosabban: a főátlóbeli aktuális elem alatti, és tőle jobbra levő almátrixban keressük az abszolút értékben legnagyobb elemet, ez lesz a *pivot-elem*.

$$\left(\begin{array}{ccccc|c} 1 & 0 & 5 & & & \\ 0 & 1 & 6 & & & \\ 0 & 0 & \mathbf{0} & \dots & \dots & \dots \\ & & & x & x & \\ & & & x & x & \end{array}\right)$$

Az x-szel jelölt elemek között keresünk új pivot elemet.

Nem bizonyított állítás, csak sejtés, hogy az algoritmus stabil, ha pivotnak mindig az almátrix abszolút értékben legnagyobb elemét választjuk.

További lehetőség: oszlopcsere

Eljárás:

A baloldali mátrix esetében:

- A megfelelő sorokat **és** oszlopokat megcseréljük, hogy a pivotelem a főátlóba kerüljön
- Feljegyezzük, hogy melyik két oszlopot cseréltük meg
- ez valójában a változók átnevezése, így a jobboldal megfelelő sorait az eljárás végén vissza kell majd cserélni!

További lehetőség: oszlopcsere

Eljárás:

A baloldali mátrix esetében:

- A megfelelő sorokat **és** oszlopokat megcseréljük, hogy a pivotelem a főátlóba kerüljön
- Feljegyezzük, hogy melyik két oszlopot cseréltük meg
- ez valójában a változók átnevezése, így a jobboldal megfelelő sorait az eljárás végén vissza kell majd cserélni!

A jobboldali mátrix esetében:

- csak a sorcserét végezzük el.

További lehetőség: oszlopcsere

Eljárás:

A baloldali mátrix esetében:

- A megfelelő sorokat **és** oszlopokat megcseréljük, hogy a pivotelem a főátlóba kerüljön
- Feljegyezzük, hogy melyik két oszlopot cseréltük meg
- ez valójában a változók átnevezése, így a jobboldal megfelelő sorait az eljárás végén vissza kell majd cserélni!

A jobboldali mátrix esetében:

- csak a sorcserét végezzük el.

Majd mindkét oldalon az aktuális sorral lefelé eliminálunk.

Ezt nevezzük *teljes pivotolásnak*.

Teljes vs részleges pivotálás

Melyiket használjuk?

- részleges pivotálás egyszerűbb, nem kell a megoldásvektor elemeinek permutálásával foglalkozni
- de csak olyan számok szolgálhatnak pivot elemként, amelyek már a megfelelő oszlopban vannak.

Általában a részleges pivotálás "majdnem" olyan jó, mint a teljes

Az algoritmus építőkövei

Azonosítjuk a fő részalgoritmusokat, amiket külön-külön könnyű megírni.

- Eldöntjük, hogy milyen formában tároljuk a mátrixokat (sorok vagy oszlopok szerint)
- Azonosítjuk az alapvető műveleteket:
  - Legnagyobb abszolút értékű elem megtalálása sorban
  - Sor szorzása (osztása) számmal
  - Két sor különbségének képzése
  - Legnagyobb abszolút értékű elem megtalálása oszlopban, főátló alatt
  - Két sor cseréje
  - Legnagyobb abszolút értékű elem megtalálása almátrixban
  - Két oszlop cseréje (oszlopcsere könyvelése)

Az algoritmus építőkövei

Azonosítjuk a fő részalgoritmusokat, amiket külön-külön könnyű megírni.

- Eldöntjük, hogy milyen formában tároljuk a mátrixokat (sorok vagy oszlopok szerint)
- Azonosítjuk az alapvető műveleteket:
  - Legnagyobb abszolút értékű elem megtalálása sorban
  - Sor szorzása (osztása) számmal
  - Két sor különbségének képzése
  - Legnagyobb abszolút értékű elem megtalálása oszlopban, főátló alatt
  - Két sor cseréje
  - Legnagyobb abszolút értékű elem megtalálása almátrixban
  - Két oszlop cseréje (oszlopcsere könyvelése)

- Az építőelemeket külön-külön megvalósítjuk és *teszteljük*!
- Alulról felfelé, az egyes függvényeket alaposan kipróbálva haladjunk!

Az algoritmus építőkövei

Azonosítjuk a fő részalgoritmusokat, amiket külön-külön könnyű megírni.

- Eldöntjük, hogy milyen formában tároljuk a mátrixokat (sorok vagy oszlopok szerint)
- Azonosítjuk az alapvető műveleteket:
  - Legnagyobb abszolút értékű elem megtalálása sorban
  - Sor szorzása (osztása) számmal
  - Két sor különbségének képzése
  - Legnagyobb abszolút értékű elem megtalálása oszlopban, főátló alatt
  - Két sor cseréje
  - Legnagyobb abszolút értékű elem megtalálása almátrixban
  - Két oszlop cseréje (oszlopcsere könyvelése)

- Az építőelemeket külön-külön megvalósítjuk és *teszteljük*!
- Alulról felfelé, az egyes függvényeket alaposan kipróbálva haladjunk!

- Ezek után jöhet a fő algoritmus leprogramozása
- Többfajta mátrixon teszteljük (0 elem a főátlóban, stb)

Kitekintés: $L\,U$ dekompozíció

Elterjedt és sokak által használt numerikus könyvtárakban, pl a LAPACKban, az ún. $L\,U$ dekompozíción alapuló lineáris egyenletrendszer megoldó algoritmus található

Előkészület: Gauss-elimináció visszahelyettesítéssel

Ez nagyjából azonos a Gauss-Jordan eliminációval, de

- Eliminálás kor csak a főátló alatti elemeket nullázuk le
- csak részleges pivotálást alkalmazunk (sorok cseréje)
- A főátlóban minden elemet 1-esre hozunk

Pl:

$$\left(\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 2 & 8 \end{array}\right) \longrightarrow \left(\begin{array}{ccc|c} \mathbf{1} & 3 & 1 & 9 \\ \mathbf{0} & \mathbf{1} & 1 & 4 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & 4 \end{array}\right)$$

Hogyan kapjuk meg a megoldást ?

Előkészület: Gauss-elimináció visszahelyettesítéssel

Ez nagyjából azonos a Gauss-Jordan eliminációval, de

- Eliminációskor csak a főátló alatti elemeket nullázuk le
- csak részleges pivotálást alkalmazunk (sorok cseréje)
- A főátlóban minden elemet 1-esre hozunk

Pl:

$$\left(\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 2 & 8 \end{array}\right) \longrightarrow \left(\begin{array}{ccc|c} \mathbf{1} & 3 & 1 & 9 \\ \mathbf{0} & \mathbf{1} & 1 & 4 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & 4 \end{array}\right)$$

Hogyan kapjuk meg a megoldást ?

Az utolsó elemet az megoldásvektorból csak le kell olvasni: $x_3 = 4$

Előkészület: Gauss-elimináció visszahelyettesítéssel

Ez nagyjából azonos a Gauss-Jordan eliminációval, de

- Eliminációskor csak a főátló alatti elemeket nullázuk le
- csak részleges pivotálást alkalmazunk (sorok cseréje)
- A főátlóban minden elemet 1-esre hozunk

Pl:

$$\left(\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 2 & 8 \end{array}\right) \longrightarrow \left(\begin{array}{ccc|c} \mathbf{1} & 3 & 1 & 9 \\ \mathbf{0} & \mathbf{1} & 1 & 4 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & 4 \end{array}\right)$$

Hogyan kapjuk meg a megoldást ?

Az utolsó elemet az megoldásvektorból csak le kell olvasni: $x_3 = 4$

A következő már egyszerű: $x_2 = 4 - 1 \cdot x_3$

Előkészület: Gauss-elimináció visszahelyettesítéssel

Ez nagyjából azonos a Gauss-Jordan eliminációval, de

- Eliminációskor csak a főátló alatti elemeket nullázuk le
- csak részleges pivotálást alkalmazunk (sorok cseréje)
- A főátlóban minden elemet 1-esre hozunk

Pl:

$$\left(\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 2 & 8 \end{array}\right) \longrightarrow \left(\begin{array}{ccc|c} \mathbf{1} & 3 & 1 & 9 \\ \mathbf{0} & \mathbf{1} & 1 & 4 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & 4 \end{array}\right)$$

Hogyan kapjuk meg a megoldást ?

Az utolsó elemet az megoldásvektorból csak le kell olvasni: $x_3 = 4$

A következő már egyszerű: $x_2 = 4 - 1 \cdot x_3$

Általában: az előző x_i-k ismeretében az x_{i-1} meghatározható:
$x_i = b_i' - \sum_{j=i+1}^{N} a_{ij}' x_j$

Előkészület: Gauss-elimináció visszahelyettesítéssel

Ez nagyjából azonos a Gauss-Jordan eliminációval, de

- Eliminációskor csak a főátló alatti elemeket nullázuk le
- csak részleges pivotálást alkalmazunk (sorok cseréje)
- A főátlóban minden elemet 1-esre hozunk

Pl:

$$\left(\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 2 & 8 \end{array}\right) \longrightarrow \left(\begin{array}{ccc|c} \mathbf{1} & 3 & 1 & 9 \\ \mathbf{0} & \mathbf{1} & 1 & 4 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & 4 \end{array}\right)$$

Hogyan kapjuk meg a megoldást ?

Az utolsó elemet az megoldásvektorból csak le kell olvasni: $x_3 = 4$

A következő már egyszerű: $x_2 = 4 - 1 \cdot x_3$

Általában: az előző x_i-k ismeretében az x_{i-1} meghatározható:
$x_i = b_i' - \sum_{j=i+1}^{N} a_{ij}' x_j$

Meg lehet mutatni, hogy elvileg ez az eljárás összességében kevesebb műveletet igényel, mint a Gauss-Jordan elimináció (ha az A^{-1}-t nem kell számolni)

Általánosabban

Ha az egyenletrendszer a következő alakra hozható:

$$\begin{pmatrix} a'_{11} & a'_{12} & a'_{13} \\ 0 & a'_{22} & a_{23} \\ 0 & 0 & a'_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ b'_3 \end{pmatrix}$$

Általánosabban

Ha az egyenletrendszer a következő alakra hozható:

$$\begin{pmatrix} a'_{11} & a'_{12} & a'_{13} \\ 0 & a'_{22} & a_{23} \\ 0 & 0 & a'_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ b'_3 \end{pmatrix}$$

akkor a megoldás:

$$\begin{aligned} x_3 &= b'_3 / a'_{33} \\ x_i &= \frac{1}{a'_{ii}} \left[b'_i - \sum_{j=i+1}^{N} a'_{ij} x_j \right] \end{aligned}$$

$L\,U$ dekompozíció

Alapötlet: tegyük fel, hogy tetszőleges $\mathbf{A}$ mátrix átírható $\mathbf{A} = \mathbf{L} \cdot \mathbf{U}$ formába!

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{31} & a_{33} \end{pmatrix} = \begin{pmatrix} 1.0 & 0 & 0 \\ l_{21} & 1.0 & 0 \\ l_{31} & l_{32} & 1.0 \end{pmatrix} \cdot \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

$L\,U$ dekompozíció

Alapötlet: tegyük fel, hogy tetszőleges $\mathbf{A}$ mátrix átírható $\mathbf{A} = \mathbf{L} \cdot \mathbf{U}$ formába!

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{31} & a_{33} \end{pmatrix} = \begin{pmatrix} 1.0 & 0 & 0 \\ l_{21} & 1.0 & 0 \\ l_{31} & l_{32} & 1.0 \end{pmatrix} \cdot \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

Ekkor az $\mathbf{Ax} = (\mathbf{L} \cdot \mathbf{U})\mathbf{x} = \mathbf{L} \cdot (\mathbf{Ux}) = \mathbf{b}$ egyenletrendszer megoldása két részre bontható:
Először keressük $\mathbf{y}$-t, amelyre

$$\mathbf{L} \cdot \mathbf{y} = \mathbf{b}$$

majd ennek ismeretében megoldjuk a

$$\mathbf{U} \cdot \mathbf{x} = \mathbf{y}$$

rendszert.

$L\,U$ dekompozíció

Először keressük $\mathbf{y}$-t, amelyre $\mathbf{L} \cdot \mathbf{y} = \mathbf{b}$ majd ennek ismeretében megoldjuk a $\mathbf{U} \cdot \mathbf{x} = \mathbf{y}$ rendszert.

Miért jó ez?

$L\,U$ dekompozíció

Először keressük **y**-t, amelyre $\mathbf{L} \cdot \mathbf{y} = \mathbf{b}$ majd ennek ismeretében megoldjuk a $\mathbf{U} \cdot \mathbf{x} = \mathbf{y}$ rendszert.

Miért jó ez? Mind a két egyenletrendszer megoldása egyszerű:

$$\begin{aligned} y_1 &= b_1 \\ y_i &= b_i - \sum_{j=1}^{i-1} l_{ij} y_j \end{aligned}$$

és

$$\begin{aligned} x_N &= y_N / u_{NN} \\ x_i &= \frac{1}{u_{ii}} \left[y_i - \sum_{j=i+1}^{N} u_{ij} x_j \right] \end{aligned}$$

$L\,U$ dekompozíció

Először keressük **y**-t, amelyre $\mathbf{L} \cdot \mathbf{y} = \mathbf{b}$ majd ennek ismeretében megoldjuk a $\mathbf{U} \cdot \mathbf{x} = \mathbf{y}$ rendszert.

Miért jó ez? Mind a két egyenletrendszer megoldása egyszerű:

$$\begin{aligned} y_1 &= b_1 \\ y_i &= b_i - \sum_{j=1}^{i-1} l_{ij} y_j \end{aligned}$$

és

$$\begin{aligned} x_N &= y_N / u_{NN} \\ x_i &= \frac{1}{u_{ii}} \left[y_i - \sum_{j=i+1}^{N} u_{ij} x_j \right] \end{aligned}$$

Hogyan határozzuk meg az **L**-t és az **U** ?

$L\,U$ dekompozíció

- Állítás: **L** és **U** hatékonyan számolható az ún. *Crout algoritmus* segítségével
- ha szükséges: részleges pivotálás a *Crout algoritmus*ban
- ha nincs szükség $\mathbf{A}^{-1}$ számolására akkor ez a leggyorsabb algoritmus

$L\,U$ dekompozíció

- Állítás: $\mathbf{L}$ és $\mathbf{U}$ hatékonyan számolható az ún. *Crout algoritmus* segítségével
- ha szükséges: részleges pivotálás a *Crout algoritmus*ban
- ha nincs szükség $\mathbf{A}^{-1}$ számolására akkor ez a leggyorsabb algoritmus

További előnyök:

- A determináns is könnyen számolható: $\det \mathbf{A} = \Pi_{i=1}^{N} u_{ii}$
- a megoldás *iteratív* pontosítása könnyen implementálható

A megoldás iteratív pontosítása

- nagy mátrixok esetén a kerekítési hibák felhalmozódhatnak még ha a mátrix nem is szinguláris
- ez a megoldás pontosságának jelentős csökkenéséhez vezethet
- az iteratív módszer segíthet a pontosság visszaszerzésében

A megoldás iteratív pontosítása

- nagy mátrixok esetén a kerekítési hibák felhalmozódhatnak még ha a mátrix nem is szinguláris
- ez a megoldás pontosságának jelentős csökkenéséhez vezethet
- az iteratív módszer segíthet a pontosság visszaszerzésében

Iteratív lépés alapgondolata:

- $\mathbf{Ax} = \mathbf{b}$ megoldását keressük
- az első lépésben kapott megoldás:$\mathbf{x} + \delta\mathbf{x}$
- $\mathbf{x}$ egzakt megoldás; $\delta\mathbf{x}$ a hiba
- mind $\mathbf{x}$ mind $\delta\mathbf{x}$ ismeretlenek

Visszahelyettesítve:

$$\mathbf{A}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b} \tag{1}$$

A megoldás iteratív pontosítása

- nagy mátrixok esetén a kerekítési hibák felhalmozódhatnak még ha a mátrix nem is szinguláris
- ez a megoldás pontosságának jelentős csökkenéséhez vezethet
- az iteratív módszer segíthet a pontosság visszaszerzésében

Iteratív lépés alapgondolata:

- $\mathbf{Ax} = \mathbf{b}$ megoldását keressük
- az első lépésben kapott megoldás:$\mathbf{x} + \delta\mathbf{x}$
- $\mathbf{x}$ egzakt megoldás; $\delta\mathbf{x}$ a hiba
- mind $\mathbf{x}$ mind $\delta\mathbf{x}$ ismeretlenek

Visszahelyettesítve:

$$\mathbf{A}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b} \tag{1}$$

$$\Rightarrow \mathbf{A}\delta\mathbf{x} = \delta\mathbf{b} \tag{2}$$

A megoldás iteratív pontosítása

Fejezzük ki (1)-ből $\delta\mathbf{b}$-t és helyettesítsük (2)-be:

$$\mathbf{A}\delta\mathbf{x} = \mathbf{A}(\mathbf{x} + \delta\mathbf{x}) - \mathbf{b}$$

A megoldás iteratív pontosítása

Fejezzük ki (1)-ből $\delta\mathbf{b}$-t és helyettesítsük (2)-be:

$$\mathbf{A}\delta\mathbf{x} = \mathbf{A}(\mathbf{x} + \delta\mathbf{x}) - \mathbf{b}$$

- Az egyenlet jobb oldalán minden tag ismert $\Rightarrow$ a fenti egyenletet megoldhatjuk $\delta\mathbf{x}$-re
- az így kapott $\delta\mathbf{x}$-t kivonjuk az első lépésben kapott $\mathbf{x} + \delta\mathbf{x}$ megoldásból

További irodalom