41. Szimmetrikus mátrixok Cholesky-féle felbontása

Benyújtja: Kaszaki Péter (KAPMAAT.SZE)
2005 november 21.

Tartalomjegyzék

1. Bevezetés

Matematikai számításaink során gyakran felmerül a lineáris egyenletrendszerek megoldása. A matematika történelme során sokan, sokféle megoldást adtak erre a problémára: lényegében kétféle módszertípus alakult ki: az egyik, amikor megpróbáljuk az egyenletekből a pontos[1] megoldást kinyerni, a másik, amikor csupán csak meg akarjuk közelíteni a megoldást, viszont minden lépéssel közelebb és közelebb szándékozunk kerülni a megoldáshoz.

Ez előbbi elv alapján működő algoritmusokat *direkt módszereknek*, míg az utóbbiakat *iterációs módszereknek* hívjuk. Mindkét módszertípusnak megvan a maga használhatósági/alkalmazási köre, amelyet az esszé egy későbbi részén részletesen elemzek.

A direkt módszerek közé sorolható a Carl Friedrich Gaussról elnevezett Gauss elimináció vagy más néven Gauss-Jordan elimináció. Ennek a módszernek a mátrixos alakban történő alkalmazása az egyenlet mátrixának LU felbontása, melynek egy speciális esete az *André-Louis Cholesky* francia matematikusról elnevezett Cholesky-féle felbontás.

Az esszé további részeiben a Cholesky-féle felbontást fogom elemezni változatos szempontok szerint.

2. A Gauss elimináció és az LU felbontás

Mielőtt még definiáljuk a Cholesky-féle felbontást, két fontos fogalmat nem árt tisztázni. Ezek a Gauss elimináció és az LU felbontás. Nem tárgyalom részletesen őket, csak annyira, amennyire a Cholesky-féle felbontás megértéséhez szükséges.

2.1 Gauss elimináció

A Gauss eliminációnak két fázisa van: (1.) egy eliminációs, amikor is lépcsős alakra hozzuk az egyenletrendszert és (2.) egy behelyettesítési, amikor is az utolsó egyenlettől kezdve kifejezzük az egyenletekből a még ismeretlen változókat a már ismert változók segítségével.

(1) Eliminációs fázis:

$$\begin{array}{ccccccccccc}
a_{1,1}\cdot x_1 & + & a_{1,2}\cdot x_2 & + & a_{1,3}\cdot x_3 & + & & + & a_{1,n}\cdot x_n & = & b_1 \\
a_{2,1}\cdot x_1 & + & a_{2,2}\cdot x_2 & + & a_{2,3}\cdot x_3 & + & \cdots & + & a_{2,n}\cdot x_n & = & b_2 \\
a_{3,1}\cdot x_1 & + & a_{3,2}\cdot x_2 & + & a_{3,3}\cdot x_3 & + & & + & a_{3,n}\cdot x_n & = & b_3 \\
 & & & & \vdots & & & & & & \vdots \\
a_{n,1}\cdot x_1 & + & a_{n,2}\cdot x_2 & + & a_{n,3}\cdot x_3 & + & \cdots & + & a_{n,n}\cdot x_n & = & b_n
\end{array}$$

A fenti egyenletrendszert sorcserékkel, 0-tól különböző skalárokkal való szorzással és az egyenletek egymáshoz adásával a lent látható alakra hozzuk:

$$\begin{array}{ccccccccccc}
a_{1,1}\cdot x_1 & + & a_{1,2}\cdot x_2 & + & a_{1,3}\cdot x_3 & + & & + & a_{1,n}\cdot x_n & = & b_1 \\
 & & a_{2,2}\cdot x_2 & + & a_{2,3}\cdot x_3 & + & \cdots & + & a_{2,n}\cdot x_n & = & b_2 \\
 & & & & a_{3,3}\cdot x_3 & + & & + & a_{3,n}\cdot x_n & = & b_3 \\
 & & & & & & & & \vdots & & \vdots \\
 & & & & & & & & a_{n,n}\cdot x_n & = & b_n
\end{array}$$

1 kerekítési és örökölt hibát leszámítva

(2.) Behelyettesítési fázis:

Az utolsó egyenletből kifejezem az utolsó ismeretlent, majd az utolsó előtti egyenletből az utolsó előtti ismeretlent, … , végül az első egyenletből az első ismeretlent, felhasználva az eddig kiszámolt ismeretlenek értékét:

$$a_{n,n}\cdot x_n = b_n \quad \rightarrow \quad x_n = \frac{b_n}{a_{n,n}}$$

$$a_{n-1,n-1}\cdot x_{n-1} + a_{n-1,n}\cdot x_n = b_n \quad \rightarrow \quad x_{n-1} = \frac{(b_n - a_{n-1,n}\cdot x_n)}{a_{n,n}}$$

$$\vdots$$

$$a_{1,1}\cdot x_1 + a_{1,2}\cdot x_2 + a_{1,3}\cdot x_3 + \cdots + a_{1,n}\cdot x_n = b_1 \quad \rightarrow \quad x_1 = \frac{(b_1 - a_{1,2}\cdot x_2 + a_{1,3}\cdot x_3 + \cdots + a_{1,n}\cdot x_n)}{a_{1,1}}$$

2.1.2 A Gauss elimináció mátrixos alakban

Lévén a számítógépek gyorsabban képesek számolni, ha az adatokat vektorok formájában tároljuk és nem egyedi változókként (Ez a processzorok gyorsítótáras felépítéséből adódik), ezért célszerűnek látszik a Gauss elimináció mátrixokkal való implementálása. A Gauss elimináció mindkét fázisát leírhatjuk mátrixműveletekkel is. Legyen:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix}, \quad \underline{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \text{ és } \quad \underline{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}.$$

Ekkor a kiindulási egyenletünk felírhatjuk $A\cdot\underline{x} = \underline{b}$ alakban is.

A Gauss elimináció első fázisában megfelelő M_i eliminációs mátrixokkal való szorzással hozzuk létre az egyenletrendszer lépcsős alakját. Mátrixos alakban felírva az egyenletünk $U\cdot\underline{x} = \underline{c}$ alakú lesz, ahol U egy felső háromszögmátrixot jelöl.

A második fázisban pedig változó behelyettesítéssel megoldjuk az $U\cdot\underline{x} = \underline{c}$ egyenletrendszert.

2.2 Az LU felbontás

Az A mátrix LU felbontásán egy olyan L és U mátrixpárost értünk, melyre teljesül, hogy $A = L\cdot U$, ahol L alsó háromszögmátrix, U pedig felső háromszögmátrix.

A Gauss elimináció LU felbontás segítségével is felírható. Ha ismerjük az egyenlet mátrixának LU felbontását, akkor a következő kettő, az eredeti egyenletrendszernél egyszerűbben megoldható egyenletrendszert kell csak megoldani:

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

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

3. A Cholesky-féle felbontás definiálása

3.1 Definíció: mátrixok szimmetriája

Egy A mátrix szimmetrikus, ha $A = A^T$. Ha egy mátrix szimmetrikus, akkor négyzetes is.

3.2 Definíció: pozitív definit mátrix

Egy A mátrix pozitív definit, ha $\underline{x}^T \cdot A \cdot \underline{x} > 0 \quad \forall \underline{x} \neq \underline{0} \in \mathbb{R}$ esetén.

3.3 Definíció: Cholesky-féle felbontás

Adott egy A szimmetrikus, pozitív definit mátrix a $\mathbb{R}^{n \times n}$ számtest felett. Az A mátrix Cholesky-féle felbontása alatt egy olyan $L \in \mathbb{R}^{n \times n}$ alsó háromszögmátrixot értünk, melyre $A = L \cdot L^T$ teljesül.

4. A Cholesky-féle felbontás tulajdonságai, alkalmazhatóságai

A Cholesky-féle felbontás az LU felbontás speciális esete. Ezért ha a Cholesky-féle felbontás egy tulajdonsága az LU felbontásra is teljesül, akkor azt a tulajdonságot az LU felbontásra mondom ki.

4.1 A Cholesky-féle felbontás egyértelmű

Tegyük fel, hogy az A mátrixnak két Cholesky-féle felbontása is létezik:

$$A = L_1 \cdot L_1^T = L_2 \cdot L_2^T$$

Ekkor felírhatjuk:

$$L_2^{-1} \cdot L_1 = L_2^T \cdot (L_2^T)^{-1}$$

Így most az egyenlet bal oldalán egy alsó háromszögmátrix áll, a jobb oldalán pedig egy felső háromszögmátrix. Ez csak úgy lehet, ha $L_2^{-1} \cdot L_1$ és $L_2^T \cdot (L_2^T)^{-1}$ is diagonális.
Legyen

$$D = L_2^{-1} \cdot L_1 = L_2^T \cdot (L_2^T)^{-1} \quad .$$

Ekkor

$$D^2 = D \cdot D^T = L_2^{-1} \cdot L_1 \cdot (L_2^T \cdot (L_2^T)^{-1})^T = L_2^{-1} \cdot (L_1 \cdot L_2^{-1}) \cdot L_2 = L_2^{-1} \cdot I \cdot L_2 = I \quad .$$

Ezzel beláttuk:

$$L_1 = L_2 \quad .$$

4.2 Mátrixok invertálása LU felbontás segítségével

Legyen X az A mátrix inverze, $\underline{x}_i$ az X i. oszlopvektora, I az egységmátrix és $\underline{e}_i$ az egységmátrix i. oszlopvektora.

Ekkor felírhatjuk:

$$A \cdot X = I$$

továbbá a következő egyenletrendszereket:

$$A \cdot \underline{x}_1 = \underline{e}_1$$
$$A \cdot \underline{x}_2 = \underline{e}_2$$
$$\vdots$$
$$A \cdot \underline{x}_n = \underline{e}_n$$

Felhasználva az A mátrix LU felbontását:

$$U \cdot \underline{x}_1 = L^{-1} \cdot \underline{e}_1$$
$$U \cdot \underline{x}_2 = L^{-1} \cdot \underline{e}_2$$
$$\vdots$$
$$U \cdot \underline{x}_n = L^{-1} \cdot \underline{e}_n$$

Ha a [8.] 3.3 fejezetében alkalmazott módszert használjuk LU felbontásra, akkor az L^{-1} -t is megkapjuk a felbontás során, így az inverz számítással nem kell bajlódnunk.

4.3 Mátrix determinánsának kiszámolása Cholesky-féle felbontás segítségével

Ha az A $n \times n$ mátrixnak létezik Cholesky-féle felbontása ($A = L \cdot L^T$), akkor a determinánsa a következő módon számolható ([1.] könyv 4.2.4 segédtétele alapján):

$$det(A) = \prod_{i=1}^{n} l_{i,i}^2$$

4.4 Sajátértékek, sajátvektorok és az LU felbontás

A sajátértékek és sajátvektorok kiszámolására is felhasználható az LU felbontás. Lássuk hogyan!

4.4.1 Sajátérték, sajátvektor definíciója

Legyen A négyzetes mátrix. λ az A mátrix sajátértéke és $\underline{x} \neq \underline{0}$ a hozzá tartozó sajátvektor, ha

$$A \cdot \underline{x} = \lambda \cdot \underline{x}$$

teljesül.

4.4.2 Karakterisztikus polinom

Az A mátrix karakterisztikus polinomján a

$$det(A - \lambda \cdot i) = 0$$

polinomot értjük. A karakterisztikus polinom gyökei az A mátrix sajátértékeit adják.

4.4.3 Sajátértékek kiszámolása az LU felbontás segítségével

Vegyük a következő iterációt:

$A_1 = A$

$A_k = U_{k-1} \cdot L_{k-1}$

ahol L_{k-1} , U_{k-1} az A_{k-1} Mátrix LU felbontása. [8.] 4.6 fejezete megmutatja, hogy A_k egy olyan mátrixhoz konvergál, amelynek a főátlójában az A mátrix sajátértékei találhatók, abszolút értékeik szerint csökkenő sorrendben.

4.4.4 Sajátvektorok kiszámítása az LU felbontás segítségével

Ha adott egy λ sajátértéke az A mátrixnak, akkor a hozzá tartozó $\underline{x}$ sajátvektort a következő lineáris egyenletrendszer megoldásával számolhatjuk ki:

$(A - \lambda \cdot I) \cdot \underline{x} = \underline{0}$

Ha ismerjük $(A - \lambda \cdot I)$ LU felbontását, akkor elég csak

$U \cdot \underline{x} = \underline{0}$ -t kiszámolni, hiszen $L^{-1} \cdot \underline{0} = \underline{0}$

4.5 Egyenletrendszerek közelítő megoldásainak iteratív finomítása LU felbontás segítségével

Az LU felbontás egyik fontos alkalmazása a pontatlan megoldás iteratív finomítása.

Tegyük fel, hogy az

(1.) $A \cdot \underline{x} = \underline{b}$

egyenletrendszerre van egy közelítő megoldásunk, jelöljük ezt $\underline{x}^{(0)}$ -lal.

Legyen $\underline{r}^{(0)}$ a maradékvektor, amelyet a következő képlet segítségével számolunk:

(2.) $\underline{r}^{(i)} = A \cdot x^{(i)} - \underline{b}$

Legyen $\underline{c}^{(i)}$ korrekció az $\underline{x}^{(i)}$ közelítő megoldás eltérése az $\underline{x}$ pontos megoldástól:

(3.) $\underline{x} = \underline{x}^{(i)} + \underline{c}^{(i)}$

(1.)-be behelyettesítve (3.)-t kapjuk:

$A \cdot (\underline{x}^{(i)} + \underline{c}^{(i)}) - \underline{b} = \underline{0}$

(2.) behelyettesítve adódik:

$A \cdot \underline{c}^{(i)} + \underline{r}^{(i)} = \underline{0}$

Ezt az egyenletrendszert $\underline{c}^{(i)}$ -re nézve megoldjuk, majd képezzük az

$\underline{x}^{(i+1)} = \underline{x}^{(i)} + \underline{c}^{(i)}$ összeget.

Az iterációt addig folytatjuk, amíg eléggé meg nem közelítjük a pontos megoldást.

5. Algoritmusok Cholesky felbontásra

A következő részben három számolási módszert mutatok be és elemzek Cholesky-féle felbontásra, ezen felül bemutatom az egyik kézi számolásnál alkalmazott változatát is. A három algoritmusból kettő csak minimális mértékben tér el, ezért együtt tárgyalom őket.

5.1 Cholesky algoritmus

A Cholesky algoritmus a Gauss elimináció egy módosított változata. Eredetileg szimmetrikus, pozitív definit mátrixú egyenletek megoldására találták ki. Ezt az algoritmust mátrixos alakba *Tadeusz Banachiewicz* lengyel matematikus írta fel.

Az algoritmus annyi lépésből áll, ahány oszlopa (sora) van a felbontandó mátrixnak. Az algoritmus minden egyes lépése a kiszámítandó L mátrix egy oszlopát adja eredményül (L_i -t).

Tekintsük most az algoritmus formális felírását:

Legyen A az a mátrix, amelynek keressük a Cholesky-féle felbontását, legyen L az A Cholesky-féle felbontása, továbbá jelöljük I_n -el az $n \times n$ -es egységmátrixot, és 0 -val a zérusmátrixot.

Legyen

$$A^{(1)} = A \quad .$$

Ha

$$A^{(i)} = \begin{pmatrix} I_{i-1} & 0 & 0 \\ 0 & a_{i,i} & \underline{b}_i^T \\ 0 & \underline{b}_i & B^{(i)} \end{pmatrix}$$

alakú, akkor $A^{(i+1)}$ -t a következő helyettesítésekkel kapjuk:

$$a_{i,i} := 1; \quad \underline{b_i} := 0; \quad \underline{b_i^T} := 0; \quad B^{(i)} := B^{(i)} \frac{1}{a_{i,i}} \underline{b}_i \underline{b}_i^T$$

Ezáltal $A^{(i+1)}$ a következő alakú lesz:

$$A^{(i+1)} = \begin{pmatrix} I_{i-1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & B^{(i)} - \frac{1}{a_{i,i}} \underline{b}_i \underline{b}_i^T \end{pmatrix}$$

$A^{(i)}$ -ből L_i -t a következő módon kapjuk:

$$L_i := \begin{pmatrix} I_{i-1} & 0 & 0 \\ 0 & \sqrt{a_{i,i}} & 0 \\ 0 & \frac{1}{\sqrt{a_{i,i}}} \underline{b}_i & I_{n-i} \end{pmatrix}$$

Miután L_n -t is kiszámoltuk L a következő módon adódik:

$$L = L_1 \cdot L_2 \cdot \ldots \cdot L_n$$

MATLAB nyelven a Cholesky algoritmus a következő kóddal valósítható meg:

```
function L = CholAlg(A)
n = size(A);
L = eye(n);
Ai = A;
for i = 1 : n
    Li = eye(n);
    Li(i,i) = sqrt( Ai(i,i) );
    Li(i+1:n, i) = 1 / Li(i,i) * Ai((i+1):n, i);
    L = L *Li;
    Ai(i+1:n, i+1:n) = Ai(i+1:n, i+1:n) -
                       1/A(i,i) * Ai(i+1:n, i)*Ai(i, i+1:n);
end
```

Lássunk most egy példát a Cholesky algoritmusra:

Számoljuk ki a következő 4×4 mátrix Cholesky-féle felbontását:

$$A = \begin{pmatrix} 25 & 5 & 15 & 5 \\ 5 & 50 & 17 & 29 \\ 15 & 17 & 22 & 11 \\ 5 & 29 & 11 & 21 \end{pmatrix}$$

1. lépés:

$$A^{(1)} = A = \begin{pmatrix} 25 & 5 & 15 & 5 \\ 5 & 50 & 17 & 29 \\ 15 & 17 & 22 & 11 \\ 5 & 29 & 11 & 21 \end{pmatrix} \quad \rightarrow \quad L_1 = \begin{pmatrix} 5 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

2. lépés:

$$A^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 49 & 14 & 28 \\ 0 & 14 & 13 & 8 \\ 0 & 28 & 8 & 20 \end{pmatrix} \quad \rightarrow \quad L_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 4 & 0 & 1 \end{pmatrix}$$

3. lépés:

$$A^{(3)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix} \quad \rightarrow \quad L_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4. lépés:

$$A^{(4)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix} \quad \rightarrow \quad L_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

Az eredmény:

$$L = L_1 \cdot L_2 \cdot L_3 \cdot L_4 = \begin{pmatrix} 5 & 0 & 0 & 0 \\ 1 & 7 & 0 & 0 \\ 3 & 2 & 3 & 0 \\ 1 & 4 & 0 & 2 \end{pmatrix}$$

5.2 A Cholesky-Banachiewicz és a Cholesky-Crout algoritmusok:

A Cholesky-Banachiewicz és a Cholesky-Crout algoritmusok csak minimális mértékben különböznek egymástól, ezért együtt tárgyalom őket:

Legyen A az az $n \times n$ -es szimmetrikus, pozitív definit mátrix, amelynek keressük a Cholesky-féle felbontását, L pedig egy olyan alsó háromszögmátrix, amely az A mátrix Cholesky-féle felbontását adja meg:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \quad , \quad L = \begin{pmatrix} l_{1,1} & 0 & \cdots & 0 \\ l_{2,1} & l_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n,1} & l_{n,2} & \cdots & l_{n,n} \end{pmatrix} \quad , \quad A = L \cdot L^T \quad .$$

A tárgyalt algoritmusok az L mátrixot elemről-elemre számolják ki a következő képletek alapján:

L főátlóbeli elemeit így számoljuk: $l_{i,i} = \sqrt{a_{i,i} - \sum_{k=1}^{i-1} l_{i,k}^2}$

L főátló alatti elemeit $(i > j)$ pedig így: $l_{i,j} = \frac{1}{l_{j,j}} \cdot \left(a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} \cdot l_{j,k} \right)$

Összesen $\frac{n \cdot (n+1)}{2}$ elemet kell kiszámolnunk. Mindkét algoritmus először az $l_{1,1}$ elemet számolja ki. A Cholesky-Banachiewicz algoritmus soronként halad lefelé az L mátrix kiszámításával, szemben a Cholesky-Crout algoritmussal, ami viszont oszlopról oszlopra halad.

MATLAB nyelven a Cholesky-Banachiewicz algoritmus a következő kóddal valósítható meg (a kód minimális megváltoztatásával a Cholesky-Crout algoritmushoz juthatunk):

```
function L = CholBanach(A)
n = size(A);
L = eye(n);
for k = 1 : n
    for j = 1 : k-1
        L(k,j) = 1 / L(j,j) * ( A(k,j) - L(k, 1:(j-1)) * L(j, 1:(j-1))' );
    end
    L(k,k) = sqrt( A(k,k) - L(k, 1:(k-1))*L(k, 1:(k-1))'  );
end
```

5.3 „parketta algoritmus”

A most bemutatásra kerülő algoritmus megegyezik a Cholesky-Crout algoritmussal.

Ha olyan eset adódna, hogy nem számítógéppel szándékozzuk a Cholesky-féle felbontást elvégezni, akkor érdemes a szemléletes „parketta algoritmust” alkalmazni. Az algoritmus neve onnan ered, hogy amikor egy szobát parkettáznak, akkor az egyes parkettaléceket mindig az előzőre merőlegesen rakják. Ennél az algoritmusnál is hasonlóan járunk el: Ha kiszámoltuk az L i. oszlopát, akkor ezzel kiszámoltuk az L^T i. sorát is:

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

Az algoritmust a következő **A** mátrixon mutatom be:

$$A = \begin{pmatrix} 9 & 6 & 3 \\ 6 & 5 & 4 \\ 3 & 4 & 21 \end{pmatrix}$$

Számoljunk akkor!

$$\begin{pmatrix} 9 & 6 & 3 \\ 6 & 5 & 4 \\ 3 & 4 & 21 \end{pmatrix} = \begin{pmatrix} l_{1,1} & 0 & 0 \\ l_{2,1} & l_{2,2} & 0 \\ l_{3,1} & l_{3,2} & l_{3,3} \end{pmatrix} \cdot \begin{pmatrix} l_{1,1} & l_{2,1} & l_{3,1} \\ 0 & l_{2,2} & l_{3,2} \\ 0 & 0 & l_{3,3} \end{pmatrix}$$

$$9 = l_{1,1} \cdot l_{1,1} \quad \rightarrow \quad l_{1,1} = 3$$

$$\begin{pmatrix} 9 & 6 & 3 \\ 6 & 5 & 4 \\ 3 & 4 & 21 \end{pmatrix} = \begin{pmatrix} 3 & 0 & 0 \\ l_{2,1} & l_{2,2} & 0 \\ l_{3,1} & l_{3,2} & l_{3,3} \end{pmatrix} \cdot \begin{pmatrix} 3 & l_{2,1} & l_{3,1} \\ 0 & l_{2,2} & l_{3,2} \\ 0 & 0 & l_{3,3} \end{pmatrix}$$

$$6 = l_{2,1} \cdot 3 \quad \rightarrow \quad l_{2,1} = 2$$

$$3 = l_{3,1} \cdot 3 \quad \rightarrow \quad l_{3,1} = 1$$

$$\begin{pmatrix} 9 & 6 & 3 \\ 6 & 5 & 4 \\ 3 & 4 & 21 \end{pmatrix} = \begin{pmatrix} 3 & 0 & 0 \\ 2 & l_{2,2} & 0 \\ 1 & l_{3,2} & l_{3,3} \end{pmatrix} \cdot \begin{pmatrix} 3 & 2 & 1 \\ 0 & l_{2,2} & l_{3,2} \\ 0 & 0 & l_{3,3} \end{pmatrix}$$

$$5 = 2 \cdot 2 + l_{2,2} \cdot l_{2,2} \quad \rightarrow \quad l_{2,2} = 1$$

$$4 = 1 \cdot 2 + l_{3,2} \cdot 1 \quad \rightarrow \quad l_{3,2} = 2$$

$$\begin{pmatrix} 9 & 6 & 3 \\ 6 & 5 & 4 \\ 3 & 4 & 21 \end{pmatrix} = \begin{pmatrix} 3 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 2 & l_{3,3} \end{pmatrix} \cdot \begin{pmatrix} 3 & 2 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & l_{3,3} \end{pmatrix}$$

$$21 = 2 \cdot 2 + 1 \cdot 1 + l_{3,3} \cdot l_{3,3} \quad \rightarrow \quad l_{3,3} = 4$$

Tehát:

$$\begin{pmatrix} 9 & 6 & 3 \\ 6 & 5 & 4 \\ 3 & 4 & 21 \end{pmatrix} = \begin{pmatrix} 3 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 2 & 4 \end{pmatrix} \cdot \begin{pmatrix} 3 & 2 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 4 \end{pmatrix}$$

5.4 Numerikus stabilitás, műveletigény

Az LU és a Cholesky-féle felbontás a direkt módszerek közé tartozik, tehát a végeredményben csak örökölt hiba és kerekítési hiba lehet, szemben az iterációs módszerekkel, ahol az eredményt képlethiba is terheli.

Az eredményben megjelenő kerekítési hiba mértéke függ a felbontandó mátrix kondíciószámától. Ha ez közel 1, akkor a mátrix jól kondicionált, tehát kevés kerekítési hibára kell számítanunk, ha viszont nagy, akkor sok kerekítési hiba várható.

Szemben az LU felbontással, a Cholesky-féle felbontás esetén nincs lehetőségünk az egyenletrendszer mátrixának kondíciószámán javítani sor és oszlopcserékkel, hiszen a Cholesky-féle felbontáshoz szimmetrikus, pozitív definit mátrix kell.

Vizsgáljuk most meg a futási időt!

E tekintetében a Cholesky-féle felbontás a jobb:

Míg az LU felbontás műveletigénye [1.] könyv szerint

$$\frac{2}{3}n^3 + O(n^2) \quad ,$$

addig a Cholesky-féle felbontás műveletigénye

$$\frac{1}{3}n^3 + O(n^2) \quad .$$

Sajnos a MATLAB újabb verziói már nem tartalmaznak a műveletigény mérésére szolgáló utasításokat, a `flops` utasítás hatására pl. a következő hibaüzenetet kapjuk:

```
>> flops
Warning: Flop counts are no longer available.
(Type "warning off MATLAB:flops:UnavailableFunction" to suppress this
warning.)
> In C:\MATLAB6p5\toolbox\matlab\elmat\flops.m at line 11
```

Ezért a következő trükk alkalmazására kényszerültem:

```
function mero(n)
A = rand(n,n);
A = A*A';
tic;
chol(A);
Chol = toc
tic;
CholBanach(A);
CholB = toc
```

```
tic;
CholAlg(A);
CholA = toc
```

A fenti függvény segítségével a következő eredményeket kaptam:

n	chol	CholBanch	CholAlg
5	0	0	0
10	0	0.0160	0
50	0	0.0310	0.0310
100	0	0.0940	0.1250
150	0	0.2030	0.5620
200	0.0160	0.3750	1.7340
300	0.0160	0.9690	9.2340
400	0.0160	1.8750	29.4840
500	0.0310	3.2650	69.9220

Ezek alapján a beépített `chol` függvény a leggyorsabb, ezután jön a `CholBanach` szkript és legvégül igen rossz futási idővel a `CholAlg` szkript. A `CholAlg` szkript azért ilyen lassú mert minden egyes lépésben ki kell számolnia az L_i mátrixon kívül az $A^{(i)}$ -t is, ez pedig sok időbe telik, ha A nagy.

6. Cholesky felbontás a MATLAB-ban

6.1 Lineáris egyenletrendszerek a MATLAB-ban

A MATLAB lineáris egyenletrendszerekkel kapcsolatos utasításai az LU, a Cholesky és az ortogonális QR felbontáson alapulnak. Ezek megvalósítására az lu, a chol, és a qr utasítások szolgálnak.

Egy $A \cdot \underline{x} = \underline{b}$ alakú lineáris egyenletrendszert így oldhatunk meg MATLAB-ban:

```
>> x = A\b
```

Ha egy egyenletrendszer mátrixa szinguláris, akkor figyelmeztetést kapunk:

```
>> A = zeros(6);
>> b=10*rand(6,1);
>> x=A\b

Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 0.000000e+000.
```

```
x =
   Inf
   NaN
   NaN
   NaN
   NaN
   NaN
```

6.2 A Cholesky-féle felbontás a MATLAB-ban

A MATLAB-ban a Cholesky-féle felbontásra az `chol` és a `cholinc` parancsok szolgálnak.

A `chol` parancs használata:

(1.) `R = chol(X)` vagy
(2.) `[R, p] = chol(X)`

Lévén csak szimmetrikus mátrixoknak létezik Cholesky-féle felbontása, a chol utasítás csak az X mátrix főátlóbeli és a főátló feletti elemeit használja. Ez viszont azt is lehetővé teszi, hogy egy nem szimmetrikus mátrixra hívjuk meg a chol utasítást. Így például előállhat a következő fura helyzet:

Egy C mátrix Cholesky-féle felbontása nem ugyanazt adja eredményül, mint a C mátrix transzponáltjának Cholesky-féle felbontása. Ilyen helyzetet mi is könnyen előállíthatunk a következő kód segítségével:

```
>> A = abs(20*rand([6,6]));
>> B = abs(20*rand([6,6]));
>> A= A*A';
>> B= B*B';
>> C = tril(A)+triu(B);
>> chol(C) - chol(C')
ans =
```

0	-1.7927	2.8426	-1.4315	-5.5724	-8.3184
0	0.5119	4.5556	2.9382	7.4017	4.5552
0	0	-2.6750	1.5230	5.2770	-3.6782
0	0	0	-0.4980	-5.7359	-0.7634
0	0	0	0	-0.8718	-6.2484
0	0	0	0	0	3.9568

Ha az X mátrix nem pozitív definit, akkor az (1.) hívási mód esetén hibaüzenetet kapunk:

```
>> A = rand(6,6);
>> chol(A)
??? Error using ==> chol
Matrix must be positive definite.
```

Ha viszont a (2.) hívási módot alkalmazzuk, akkor nem kapunk hibaüzenetet:

```
>> A = rand(6,6);
>> [R, p] = chol(A)
```

```
R =

    0.9157    0.7585
         0    0.2144

p =

     3
```

Ez a hívási mód egy mátrixot (R) és egy számot (p) ad eredményként vissza.

Ha a paraméterként kapott mátrix pozitív definit, akkor az R annak Cholesky-féle felbontása, a p pedig 0 lesz.

Ha a paraméterként kapott mátrix nem pozitív definit, akkor az R mátrix a következő utasítással képezhető legnagyobb pozitív definit mátrixnak a Cholesky-féle felbontása lesz, p pedig annak mérete +1:

```
>> A(1:p-1,1:p-1)
```

A `cholinc` parancs használata:

```
R = chol(X, tol)
```

Ahol `X` a felbontandó mátrix, `tol` pedig egy toleranciaérték.

A `cholinc` parancs ritka mátrixok nem teljes, azaz közelítő (incomplete) Cholesky-féle felbontására szolgál. Ezt a parancsot ritka mátrixok iterációs módszerei kapcsán alkalmazzák prekondícionálásra.

7. Összegzés

Egy A mátrix Cholesky-féle felbontása alatt egy L alsó háromszögmátrixot értünk, amelyre

$$A = L \cdot L^T$$

teljesül.

A Cholesky-féle felbontást főleg lineáris egyenletrendszerek megoldására használják. Ezen felül lehet vele mátrixot invertálni, determinánst számolni, sajátértéket, sajátvektort meghatározni és lineáris egyenletrendszer megoldását pontosítani. Fontos szerepet játszik továbbá a statisztika területén és a Monte Carlo eljárásokban, amelyek különböző matematikai és fizikai rendszerek viselkedésének modellezésére szolgálnak.

Kiszámolni a Cholesky algoritmus, Cholesky-Banachiewicz és a Cholesky-Crout algoritmusok segítségével tudjuk.

A Cholesky-féle felbontás közel kétszer olyan gyors, mint az LU felbontás. Viszont nagyméretű mátrixok esetén már a futásidőt is nagy, nem beszélve a memóriaigényről. Ezért inkább kis és közepes méretű mátrixok, valamint nagyméretű sávmátrixok felbontására célszerű alkalmazni.

Hibák tekintetében is jól vizsgázik: numerikusan stabilis eljárás, természetesen a kiszámítandó mátrix kondicionáltságától függően.

8. Irodalomjegyzék

Könyvek:				
	Szerző(k)	**Cím**	**Kiadó**	**Kiadás**
1.	Virágh János	Numerikus matematika	JATE Press	2003
2.	Galántai Aurél, Jeney András	Numerikus módszerek	Miskolci egyetemi kiadó	1998
3.	Bálint Elemér	Közelítő matematikai módszerek	Műszaki könyvkiadó	1966
4.	Peter Henrici	Numerikus analízis	Műszaki könyvkiadó	1985
5.	Josef Stoer	Numerische Mathematik 1.	Springer-Lehrbuch	1994
6.	Josef Stoer	Numerische Mathematik 2.	Springer-Lehrbuch	1990
7.	Szabó László	Lineáris Algebra	Polygon könyvtár	2003
Internetes források, segédeszközök:				
	Megnevezés	**Cím**		
8.	A tárgy hivatalos jegyzete	www.inf.u-szeged.hu/~csendes/koszi.ps.gz		
9.	Google kereső	www.google.com		
10.	WikiPedia lexikon	www.wikipedia.org		
11.	ABSOLUTE astronomy	www.absoluteastronomy.com/Reference.htm		