ulm university universität uulm

Prof. Dr. Stefan Funken
M.Sc. Andreas Bantle
Dipl.-Math. oec. Klaus Stolle

Universität Ulm
Institut für Numerische Mathematik
Wintersemester 2014/2015

Numerische Lineare Algebra - Matlab-Blatt 3
Lösung

(Besprechung in den Matlab-Tutorien in KW 47/48)

Hinweise

Die Hinweise zur Abgabe der Übungsblätter finden Sie auf dem ersten Übungsblatt!

Aufgabe 6 *(LR-Zerlegung mit Pivotisierung)* (0+2+4+2 Punkte)

Bearbeiten Sie die folgenden Aufgaben:

(i) Laden Sie sich die Matlab-Funktion `myLUCols.m` von der Homepage, die die LR-Zerlegung einer Matrix $A \in \mathbb{R}^{n\times n}$ mit **Spalten**-Pivotisierung berechnet.

(ii) Modifizieren Sie die Matlab-Funktion `myLUCols.m`, sodass die LR-Zerlegung (L unipotent) mit **Zeilen**-Pivotisierung berechnet wird. Nennen Sie die neue Funktion `myLURows` (Aufruf: `[L,R,P]=myLURows(A)`)

(iii) Schreiben Sie eine Matlab-Funktion `[L,R,PR,PC]=myLUAbs(A)`, die die LR-Zerlegung mit absoluter Pivotisierung berechnet, d.h. das Pivotelement in der kompletten Rest-Matrix sucht. Zurückgegeben werden sollen neben den Matrizen L und R die Permutationsmatrizen PR für die Zeilen und PC für die Spalten, d.h. $PR{\cdot}A{\cdot}PC = L{\cdot}R$.

(iv) Laden Sie sich das Skript `testLR.m` von der Homepage und vervollständigen Sie die mit Kommentaren gekennzeichneten Stellen:

- Zeile 10-14: Überprüfen Sie die LR-Zerlegung mit Zeilen-Pivotisierung
- Zeile 16-20: Überprüfen Sie die LR-Zerlegung mit absoluter Pivotisierung

(Hinweis: Sie müssen nur die Zeilen 4-8 in geeigneter Weise modifizieren.)

Lösung:

(i) Function `myLUCols.m`

```
function [L,R,P] = myLUCols(A)

n = size(A,1);
p=1:n;

for k=1:n-1
  %*** bestimme maximalen Wert in k-ter Spalte
  [~,mptr] = max(abs(A(p(k:end),k)));
  tmp = p(k);
  %*** tausche Zeilen
  p(k) = p(k-1+mptr);
  p(k-1+mptr) = tmp;
  %*** f hre LR-Schritt aus
  for j = k+1 : n
    A(p(j),k) = A(p(j),k)/A(p(k),k);
    for i = k+1 : n
      A(p(j),i) = A(p(j),i) - A(p(j),k)*A(p(k),i);
    end
```

```
19    end
20  end
21  %*** extrahiere L und R
22  L = tril(A(p,:),-1)+eye(n);
23  R = triu(A(p,:));
24  %*** berechne P
25  P=eye(n);
26  P(:,p) = P;
```

(ii) Funktion `myLURows.m`

```
1   function [L, R, P] = myLURows(A)
2
3   n = size(A,1);
4   p=1:n;
5
6   for k=1:n-1
7     %*** bestimme maximalen Wert in k-ter Zeile
8     [~,mptr] = max(abs(A(k,p(k:end))));
9     tmp = p(k);
10    %*** tausche Spalten
11    p(k) = p(k-1+mptr);
12    p(k-1+mptr) = tmp;
13    %*** f hre LR-Schritt aus
14    for j = k+1 : n
15      A(j,p(k)) = A(j,p(k))/A(k,p(k));
16      for i = k+1 : n
17        A(j,p(i)) = A(j,p(i)) - A(j,p(k))*A(k,p(i));
18      end
19    end
20  end
21  %*** extrahiere L und R
22  L = tril(A(:,p),-1)+eye(n);
23  R = triu(A(:,p));
24  %*** berechne P
25  P=eye(n);
26  P(p,:) = P;
```

Im Vergleich zur Funktion `myLUCols.m` muss Folgendes angepasst werden:

- Zeile 8: Der maximale Eintrag wird in der k-ten Zeile gesucht.
- Zeile 14-18: Die Spalten werden im Algorithmus vertauscht, d.h. der Permutationsvektor steht nun bei den Spaltenindizes.
- Zeile 22-26: Die Matrizen L, R und P werden extrahiert. auch hier steht der Permutationsvektor bei den Spalten.

(iii) Funktion `myLUAbs.m` Im Vergleich zu den vorherigen Routinen muss Folgendes angepasst werden:

- Zeile 4-5: Es gibt zwei Permutationsvektoren (für Zeilen und Spalten)
- Zeile 9-10: Der maximale Eintrag wird in der Restmatrix gesucht.
- Zeile 12-19: Die Einträge der Zeilen- und Spaltenindizes des maximalen Elements werden in beiden Permutationsvektoren ausgetauscht.
- Zeile 14-18: Die Spalten und Zeilen werden im Algorithmus vertauscht, d.h. die Permutationsvektoren steht nun bei den Spalten- und Zeilenindizes.
- Zeile 22-26: Analog zu den vorherigen Routinen werden die Matrizen L und R extrahiert, und PC und PR erzeugt.

```
function [L,R,PR,PC] = myLUAbs(A)

n = size(A,1);
p1=1:n; % Zeilen-Permutation
p2=1:n; % Spalten-Permutation

for k=1:n-1
  %*** bestimme maximalen Wert in Restmatrix
  m = max(max(abs( A(p1(k:end),p2(k:end)) )));
  [r,c] = find(abs(A(p1(k:end),p2(k:end)))==m);

  %*** tausche Zeile
  tmp1 = p1(k);
  p1(k) = p1(k-1+r);
  p1(k-1+r) = tmp1;
  %*** tausche Spalte
  tmp2 = p2(k);
  p2(k) = p2(k-1+c);
  p2(k-1+c) = tmp2;
  %*** f hre LR-Schritt aus
  for j = k+1 : n
    A(p1(j),p2(k)) = A(p1(j),p2(k))/A(p1(k),p2(k));
    for i = k+1 : n
      A(p1(j),p2(i)) = A(p1(j),p2(i)) - A(p1(j),p2(k))*A(p1(k),p2(i));
    end
  end
end
%*** extrahiere L und R
L = tril(A(p1,p2),-1)+eye(n);
R = triu(A(p1,p2));
%*** berechne PR (Zeilen-Permutation),PC (Spalten-Permutation)
PR=eye(n);
PR(:,p1) = PR;
PC=eye(n);
PC(p2,:) = PC;
```

(iv) Skript `testLR.m`

```
clc, clear all, close all

%*** Teste die Funktionen auf Richtigkeit
A = rand(10);
[L,R,P] = myLUCols(A);
if max(max(abs( P*A-L*R )))<1e-14
  disp('LR-Zerlegung mit Spalten-Pivotisierung........ ok!')
end

[L,R,P] = myLURows(A);
if max(max(abs( A*P-L*R )))<1e-14
  disp('LR-Zerlegung mit Zeilen-Pivotisierung......... ok!')
end

[L,R,PR,PC] = myLUAbs(A);
if max(max(abs( PR*A*PC-L*R )))<1e-14
  disp('LR-Zerlegung mit absoluter Pivotisierung....... ok!')
end
```

Aufgabe 7 *(Cholesky-Zerlegung für Hermitesche Matrizen)* (2+5+5+0 Punkte)

Sei $A := (a_{j,k})_{j,k=1,..,n} \in \mathbb{C}^{n\times n}$ eine hermitesche Matrix ($A = A^H := \overline{A}^T$). Um komplexe Arithmetik zu vermeiden, kann die Matrix A ohne zusätzlichen Speicherplatz wie folgt gespeichert werden:

$$\begin{pmatrix} a_{11} & \mathrm{Im}\,(a_{21}) & \dots & \mathrm{Im}\,(a_{n1}) \\ \mathrm{Re}\,(a_{21}) & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathrm{Im}\,(a_{n,n-1}) \\ \mathrm{Re}\,(a_{n1}) & \dots & \mathrm{Re}\,(a_{n,n-1}) & a_{nn} \end{pmatrix}. \tag{1}$$

Um auch bei Vektoren komplexe Arithmetik zu vermeiden, speichern wir diese folgendermaßen ab:

$$\begin{pmatrix} \mathrm{Re}\,(x_1) & \mathrm{Im}\,(x_1) \\ \vdots & \vdots \\ \mathrm{Re}\,(x_n) & \mathrm{Im}\,(x_n) \end{pmatrix}.$$

Bearbeiten Sie die folgenden Aufgaben **ohne** komplexe Arithmetik zu verwenden:

(i) Schreiben Sie eine MATLAB-Funktion `y=myMatVec(A,x)`, die zu einem Vektor x und einer Matrix A in den oben genannten Formaten das Matrix-Vektor-Produkt berechnet.

(ii) Schreiben Sie eine MATLAB-Funktion `L=myCholesky(A)` die für eine positiv definite, hermitesche Matrix im Format (1) eine Cholesky-Zerlegung $A = LL^H$ mit

$$L = (\ell_{jk})_{j,k=1,\dots,n} \in \mathbb{C}^{n\times n}, \quad \ell_{j,k} = 0 \text{ für } k > j \quad \text{und} \quad \ell_{kk} \in \mathbb{R}$$

berechnet. Verwenden sie dabei den Algorithmus des ersten Theorieblattes Aufgabe 3:

$$\ell_{kk} := \sqrt{a_{kk} - \sum_{j=1}^{k-1} |\ell_{kj}|^2} \in \mathbb{R}$$

$$\ell_{ik} := \frac{1}{\ell_{kk}} \left(a_{ik} - \sum_{j=1}^{k-1} \ell_{ij}\overline{\ell_{kj}} \right).$$

Speichern Sie die Matrix L im Format (1) ab.

(iii) Schreiben Sie eine Routine `x=mySolve(A,b)`, die das LGS $Ax = b$ mit der Cholesky-Zerlegung und Vorwärts-/Rückwärtseinsetzen löst. Die Matrix A und die Vektoren x und b sind dabei wieder in oben genannten Formaten gegeben.

(iv) Laden sie sich das MATLAB-Skript `testCholesky.m` von der Homepage und testen Sie Ihre Programme.

Lösung:

(i) Function `myMatVec.m`

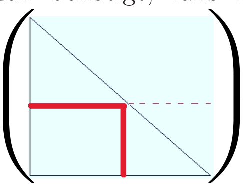
```
function y = myMatVec(A,x)
n = size(A,1);
y= zeros(n,2);

for j=1:n
  %*** berechne Realteil von y
  y(j,1) = A(j,1:j-1)*x(1:j-1,1)-A(1:j-1,j)'*x(1:j-1,2)+A(j,j)*x(j,1)+...
           A(j+1:end,j)'*x(j+1:end,1)+A(j,j+1:end)*x(j+1:end,2);
  %*** berechne Realteil von y
  y(j,2) = A(j,1:j-1)*x(1:j-1,2)+A(1:j-1,j)'*x(1:j-1,1)+A(j,j)*x(j,2)+...
           A(j+1:end,j)'*x(j+1:end,2)-A(j,j+1:end)*x(j+1:end,1);
end
```

Sei $b = A \cdot x$. Für hermitesche Matrizen gilt:

$$b_j \;=\; \sum_{k=1}^{n} a_{jk} \cdot x_k \;=\; \sum_{k=1}^{j-1} a_{jk} \cdot x_k \; + a_{jj} \cdot x_j \; + \sum_{k=j+1}^{n} \overline{a_{kj}} \cdot x_k$$

Folgende Einträge der Matrix A werden benötigt, falls nur der symmetrische Anteil gespeichert wird:

j-te Zeile $\rightarrow$ 

$\Rightarrow$ mit reeller Arithmetik ergibt sich:

$$\begin{aligned}
b_j \;=& \sum_{k=1}^{j-1} \left([\mathrm{Re}\,(a_{jk}) \cdot \mathrm{Re}\,(x_k) - \mathrm{Im}\,(a_{jk}) \cdot \mathrm{Im}\,(x_k)] \;+\; i \cdot [\mathrm{Re}\,(a_{jk}) \cdot \mathrm{Im}\,(x_k) + \mathrm{Im}\,(a_{jk}) \cdot \mathrm{Re}\,(x_k)] \right) \\
&+\; [\mathrm{Re}\,(a_{jj}) \cdot \mathrm{Re}\,(x_j) - \mathrm{Im}\,(a_{jj}) \cdot \mathrm{Im}\,(x_j)] \;+\; i \cdot [\mathrm{Re}\,(a_{jj}) \cdot \mathrm{Im}\,(x_j) - \mathrm{Im}\,(a_{jj}) \cdot \mathrm{Re}\,(x_j)] \\
&+ \sum_{k=j+1}^{n} \left([\mathrm{Re}\,(a_{kj}) \cdot \mathrm{Re}\,(x_k) + \mathrm{Im}\,(a_{kj}) \cdot \mathrm{Im}\,(x_k)] \;+\; i \cdot [\mathrm{Re}\,(a_{kj}) \cdot \mathrm{Im}\,(x_k) - \mathrm{Im}\,(a_{kj}) \cdot \mathrm{Re}\,(x_k)] \right).
\end{aligned}$$

Man beachte, dass für $j > k$ gilt:

$$\begin{aligned}
\mathrm{Re}\,(a_{jk}) &= A(j,k) \\
\mathrm{Im}\,(a_{jk}) &= A(k,j).
\end{aligned}$$

(ii) Funktion `myCholesky.m`

```
function L = myCholesky(A)

n = size(A,1);
%*** initialisiere L
L = zeros(n);
for k=1:n
  %*** berechne Diagonal-Element (reell)
  L(k,k) = sqrt(A(k,k)- sum(L(k,1:k-1).^2 + L(1:k-1,k)'.^2));
  for i=k+1:n
    %*** berechne Realteil von L_{i,k}
    L(i,k) = (A(i,k) - (L(i,1:k-1)*L(k, 1:k-1)'+ L(1:k-1,i)'*L(1:k-1,k)))/L(k,k);
    %*** berechne Imaginaerteil von L_{i,k}=>L_{k,i}
    L(k,i) = (A(k,i) - (L(1:k-1,i)'*L(k, 1:k-1)'- L(i,1:k-1)*L(1:k-1,k)))/L(k,k);
  end
end
```

Es ergibt sich:

$$
\begin{aligned}
\ell_{ik} &= \frac{1}{\ell_{kk}} \cdot \ \left(\mathrm{Re}\,(a_{ik}) + i \cdot \mathrm{Im}\,(a_{ik}) - \sum_{j=1}^{k-1}([\mathrm{Re}\,(\ell_{ij})\mathrm{Re}\,(\overline{\ell_{kj}}) - \mathrm{Im}\,(\ell_{ij})\mathrm{Im}\,(\overline{\ell_{kj}})]\right. \\
&\quad \left. + i \cdot [\mathrm{Re}\,(\ell_{ij}) Im(\overline{\ell_{kj}}) - \mathrm{Im}\,(\ell_{ij})\mathrm{Re}\,(\overline{\ell_{kj}})])\right) \\
&= \frac{1}{\ell_{kk}} \cdot \ \left(\mathrm{Re}\,(a_{ik}) - \sum_{j=1}^{k-1}[\mathrm{Re}\,(\ell_{ij})\mathrm{Re}\,(\ell_{kj}) + \mathrm{Im}\,(\ell_{ij})\mathrm{Im}\,(\ell_{kj})]\right. \\
&\quad \left. + i \cdot [\mathrm{Im}\,(a_{ik}) - \sum_{j=1}^{k-1}(\mathrm{Re}\,(\ell_{ij})\mathrm{Im}\,(\ell_{kj}) + \mathrm{Im}\,(\ell_{ij})\mathrm{Re}\,(\ell_{kj}))]\right)
\end{aligned}
$$

(iii) Funktion `mySolve.m`

```
function x = mySolve(A,b)
n = size(A,1);
%*** Berechne Cholesky-Zerlegung
L = myCholesky(A);

%*** Vorwaertseinsetzen
y = zeros(n,2);
for j=1:n
  %*** berechne Realteil von y_j
  y(j,1) = 1/L(j,j)*(b(j,1)- (L(j,1:j-1)*y(1:j-1,1) - L(1:j-1,j)'*y(1:j-1,2)));
  %*** berechne Imaginaerteil von y_j
  y(j,2) = 1/L(j,j)*(b(j,2)- (L(j,1:j-1)*y(1:j-1,2) + L(1:j-1,j)'*y(1:j-1,1)));
end

%*** Rueckwaertseinsetzen
x = zeros(n,2);
for j=n:-1:1
  %*** berechne Realteil von y_j
  x(j,1) = 1/L(j,j)*(y(j,1)- (L(j+1:n,j)'*x(j+1:n,1) + L(j,j+1:n)*x(j+1:n,2)));
  %*** berechne Imaginaerteil von y_j
  x(j,2) = 1/L(j,j)*(y(j,2)- (L(j+1:n,j)'*x(j+1:n,2) - L(j,j+1:n)*x(j+1:n,1)));
end
```

Es ergibt sich beim Vorwärtseinsetzen (Rückwärts analog):

$$
\begin{aligned}
y_j &= \frac{1}{l_{jj}} \cdot (b_j - \sum_{k=1}^{j-1} \ell_{jk} y_k) \\
&= \frac{1}{l_{jj}} \cdot (\mathrm{Re}\,(b_j) + i \cdot \mathrm{Im}\,(b_j) - \sum_{k=1}^{j-1}(\mathrm{Re}\,(\ell_{jk}) + i \cdot \mathrm{Im}\,(\ell_{jk})) \cdot (\mathrm{Re}\,(y_k) + i \cdot \mathrm{Im}\,(y_k)) \\
&= \frac{1}{\ell_{jj}} \cdot (\mathrm{Re}\,(b_j) - \sum_{k=1}^{j-1}[\mathrm{Re}\,(\ell_{jk})\mathrm{Re}\,(y_k) - \mathrm{Im}\,(\ell_{jk})\mathrm{Im}\,(y_k)] \\
&\quad + i \cdot \mathrm{Im}\,(b_j) - i \cdot \sum_{k=1}^{j-1}[\mathrm{Im}\,(\ell_{jk})\mathrm{Re}\,(y_k) + \mathrm{Re}\,(\ell_{jk})\mathrm{Im}\,(y_k)])
\end{aligned}
$$

(iv) Skript `testCholesky.m`

```
clc,clear all, close all

%*** Definiere A, b, x
A = [       9,       3-3i,       -6-9i,      -4.5+6i,         6-3i;...
          3+3i,          6,       13-4i,     0.5-8.5i,         1-5i;...
         -6+9i,      13+4i,        55.5,   26.75-34i,   -17.25-12i;...
       -4.5-6i,  0.5+8.5i,   26.75+34i,         96.5,  -22.5-18.5i;...
         6+3i ,       1+5i,  -17.25+12i,  -22.5+18.5i,        78.25];
x = [3+1i;2+2i;-1-3i;2+5i;3-1i];
b = [1+2i;2+8i;-1-1i;1-3i;3-2i];

%*** konvertiere in reelles Format
A2 = imag(tril(A,-1))'+real(tril(A,-1))+diag(diag(A));


%*** Matrix-Vektor-Multiplikation
y = myMatVec(A2,[real(x),imag(x)]);
% Ueberpruefe Ergebnis
if max(abs(complex(y(:,1),y(:,2))-A*x))<1e-10
  disp('myMatVec..... ok!')
end

%*** LGS loesen
x = mySolve(A2,[real(b),imag(b)]);
x2 = A\b;
% Ueberpruefe Ergebnis
if max(abs(complex(x(:,1),x(:,2))-x2))<1e-10
  disp('mySolve..... ok!')
end

%*** berechne Cholesky-Zerlegung mit reellem Algorithmus
L = myCholesky(A2);
%*** konvertiere ins Standard-Format
LL = diag(diag(L))+1i*triu(L,1)'+tril(L,-1);

%*** Ueberpruefe Ergebnis
if max(max(abs(A-LL*LL')))<1e-10
  disp('Cholesky..... ok!')
end
```