Programmieren der Cholesky-Zerlegung

Bevor man mit dem Programmieren einer Aufgabe beginnen kann, muss man das Problem durch und durch verstanden haben und am besten anhand von einem kleinen Beispiel durchgerechnet haben. Erst dann kann man mit dem Programmieren beginnen. Daher wird hier genauso vorgegangen - zunächst soll erläutert werden, was die Cholesky-Zerlegung ist, dann wird ein Beispiel durchgerechnet und anschließend ein Matlab-Code hergeleitet.

Satz 1 *Sei A positiv definite $n \times n$ Matrix. Dann existiert genau eine linke untere Dreiecksmatrix L mit positiven Diagonalelementen, so dass*

$$A = LL^* \tag{1}$$

gilt. (1) heißt Cholesky-Zerlegung und ist in $\frac{1}{6}n^3 + \mathcal{O}(n^2)$ Operationen berechenbar. Lineare Gleichungssysteme $Ax = LL^*x = b$ werden dann durch Lösung der Systeme $Ly = b$ (Vorwärtseinsetzen) und $L^*x = y$ (Rückwärtseinsetzen) berechnet.

Mit der Cholesky-Zerlegung können große Gleichungssysteme sehr schnell gelöst werden, falls die Matrix A positiv definit ist. Allgemein gilt:

$$\begin{aligned} A &= \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \\ &= \begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix} \begin{pmatrix} l_{11} & \bar{l}_{21} & \cdots & \bar{l}_{n1} \\ 0 & l_{22} & \cdots & \bar{l}_{n2} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & l_{nn} \end{pmatrix} \\ &= LL^* \end{aligned}$$

Vergleichen wir A mit LL^* so ergibt sich:

$$\begin{aligned} a_{11} &= |l_{11}|^2 && \Rightarrow l_{11} = \sqrt{a_{11}} \\ a_{21} &= l_{21}l_{11} && \Rightarrow l_{21} = a_{21}/l_{11} \\ a_{22} &= |l_{21}|^2 + |l_{22}|^2 && \Rightarrow l_{22} = \sqrt{a_{22} - |l_{21}|^2} \\ a_{31} &= l_{31}l_{11} && \Rightarrow l_{31} = a_{31}/l_{11} \\ a_{32} &= l_{31}\bar{l}_{21} + l_{32}l_{22} && \Rightarrow l_{32} = (a_{32} - l_{31}\bar{l}_{21})/l_{22} \\ a_{33} &= |l_{31}|^2 + |l_{32}|^2 + |l_{33}|^2 && \Rightarrow l_{33} = \sqrt{a_{33} - |l_{31}|^2 - |l_{31}|^2} \\ &\vdots && \vdots \end{aligned}$$

Es lassen sich die Formeln

$$\begin{aligned} l_{jj} &= \sqrt{a_{jj} - |l_{j1}|^2 - \cdots - |l_{j,j-1}|^2} \\ &= \left(a_{jj} - \sum_{k=1}^{j-1} |l_{j,k}|^2 \right)^{1/2} \qquad (2) \\ l_{ij} &= (a_{ij} - l_{i1}\bar{l}_{j1} - \cdots - l_{i,j-1}\bar{l}_{j,j-1})/l_{jj} \\ &= \frac{1}{l_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{i,k}\bar{l}_{j,k} \right) \qquad (3) \end{aligned}$$

herleiten (siehe auch Skript). Formel (2) und (3) sind die Grundlage für das Cholesky-Programm. Anhand eines Beispiels untersuchen wir, in welcher Reihenfolge die Formeln (2) und (3) angewandt werden müssen.

Beispiel:

Es sei das lineare Gleichungssystem $Ax = b$ mit

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 5 & 2 \\ 1 & 2 & 10 \end{pmatrix} \qquad b = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

gegeben. Beginnen wir mit dem ersten Element der Matrix L, also mit Formel (2). Für $j = 1$ gilt

$$l_{11} = \left(a_{11} - \sum_{k=1}^{0} |l_{1,k}|^2 \right)^{1/2} = \sqrt{a_{11}}.$$

l_{22} können wir noch nicht berechnen, dazu benötigt man den Wert l_{21}, den wir noch nicht kennen. Also können wir mit Formel (2) nicht mehr weiterrechnen. Daher nutzen wir nun Formel (3). Diese gilt für den Fall $i \neq j$, also beginnen wir mit $j = 1$ und $i = 2$:

$$l_{21} = \frac{1}{l_{11}} \left(a_{21} - \sum_{k=1}^{0} l_{2,k}\bar{l}_{1,k} \right) = \frac{a_{21}}{l_{11}}.$$

Die Summe $\sum_{k=1}^{j-1} l_{i,k}\bar{l}_{j,k}$ ist wieder gleich Null. Diesen Wert wird sie für alle $i = 2, \ldots, n$ annehmen, wenn $j = 1$ ist. Also können wir die Matrixeinträge l_{i1} für alle $i = 2, \ldots, n$ mit Formel (3) berechnen:

$$l_{i1} = \frac{1}{l_{11}} \left(a_{i1} - \sum_{k=1}^{0} l_{i,k}\bar{l}_{1,k} \right) = \frac{a_{i1}}{l_{11}}.$$

Für $j = 2$ kommen wir mit Formel (3) nicht weiter, da l_{22} noch unbekannt ist. Also gehen wir wieder zu Formel (2) über und untersuchen den Fall $j = 2$:

$$l_{jj} = \left(a_{22} - \sum_{k=1}^{2-1} |l_{2,k}|^2 \right)^{1/2} = \left(a_{22} - |l_{2,1}|^2 \right)^{1/2}$$

Mehr können wir mit dieser Formel nun wieder nicht berechnen, für den Fall $j = 3$ fehlt uns $l_{3,1}$. Also gehen wir wieder zu Formel (3) über und setzen dort $j = 2$:

$$l_{i2} = \frac{1}{l_{22}} \left(a_{i2} - \sum_{k=1}^{1} l_{i,k} \bar{l}_{2,k} \right) = \frac{1}{l_{22}} \left(a_{i2} - l_{i,1} \bar{l}_{2,1} \right)$$

Die $l_{i,1}$ sind für alle $i = 1, \ldots, n$ bekannt, daher können wir l_{i2} wieder für alle i berechnen. In unseren ersten Berechnungen haben wir l_{12} und l_{22} jedoch schon berechnet, so dass wir nun erst bei $i = j + 1 = 3$ starten müssen, um die restlichen $l_{i,1}$ zu erhalten:

$$l_{i2} = \frac{1}{l_{22}} \left(a_{i2} - l_{i,1} \bar{l}_{2,1} \right), \qquad i = 3, \ldots, n.$$

Nun berechnen wir wieder mit Formel (2) das Hauptdiagonalelement l_{33} und anschließend mit Formel (3) die Elemente $l_{i,3}$. Da wiederum die Elemente $l_{1,3}$, $l_{2,3}$ und $l_{3,3}$ bekannt sind, starten wir in Formel (3) mit $i = j + 1 = 4$

$$l_{i3} = \frac{1}{l_{33}} \left(a_{i3} - \sum_{k=1}^{2} l_{i,k} \bar{l}_{3,k} \right) = \frac{1}{l_{33}} \left(a_{i3} - l_{i,1} \bar{l}_{3,1} - l_{i,2} \bar{l}_{3,2} \right).$$

Die Einträge $l_{i,1}$ und $l_{i,2}$ sind nach den ersten Berechnungen bekannt, also kann die Summe ausgewertet werden.

Nach diesem Schema läuft die ganze Berechnung ab. Wir fassen zusammen und formulieren einen ersten Pseudocode:

```
berechne zuerst l_jj mit Formel 1
berechne dann fuer i = j+1,...,n l_ij mit Formel 2
```

Diese beiden Schritte führen wir für alle $j = 1, \ldots, n$ durch. Somit ergibt sich der Pseudocode

```
fuer j=1,...,n
   berechne zuerst l_jj mit Formel 1
   berechne dann fuer i = j+1,...,n l_ij mit Formel 2
ende
```

Nun muss dieser Code in Matlab übersetzt werden. Dazu initialisieren wir zunächst die Matrix L mit dem Befehl `L=zeros(n,n)`. Die Matrixeinträge (l_{ij}) werden nun mit $L(i,j)$ angesprochen. Die Programmierung der äußeren j-Schleife ist klar:

```
for j=1:n

end
```

Zunächst wir das Element l_{jj} errechnet, wobei die Summe in einer Extra-Schleife berechnet wird. Die Summe in Formel (2) kann z.B. auf die folgende Weise programmiert werden. Wir definieren eine Hilfsvariable `summe1 = 0`, die zunächst den Wert Null hat. Dann schreiben eine Schleife über den Summationsindex k, in jedem Durchlauf wird der alte Wert der Summe durch Addition des alten Wertes mit dem neuen Summand ersetzt:

```
summe1 = 0;
for k=1:j-1
    summe1 = summe1 + (L(j,k))^2;
end
```

Wir fügen die äußere j-Schleife, diese Summenberechnung und die Berechnung des Wertes l_{jj} mit Formel (2) zusammen:

```
for j=1:n
   summe1 = 0;
   for k=1:j-1
       summe1 = summe1 + (L(j,k))^2;
   end
   L(j,j)=sqrt(A(j,j)-summe1);

end
```

Nun müssen noch die Elemente l_{ij} berechnet werden. Dies wird für ein festes j nach der Berechnung des Elements l_{jj} für $i = j+1, \ldots, n$ durchgeführt. Dementsprechend entsteht eine innere `for`-Schleife über `i=j+1:n`:

```
for j=1:n
   summe1 = 0;
   for k=1:j-1
       summe1 = summe1 + (L(j,k))^2;
   end
   L(j,j)=sqrt(A(j,j)-summe1);
```

```
   for i = j+1:n


   end
end
```

Auch in dieser Schleife muss eine Summe über k berechnet werden, dazu initialisieren wir eine Hilfsvariable `summe2 = 0`. Die Programmierung der zweiten Summe $\sum_k^{j-1} l_{i,k}\bar{l}_{j,k}$ verläuft analog zur ersten Summe:

```
for j=1:n
   summe1 = 0;
   for k=1:j-1
       summe1 = summe1 + (L(j,k))^2;
   end
   L(j,j)=sqrt(A(j,j)-summe1);
   for i = j+1:n
       summe2 = 0;
       for k=1:j-1
           summe2 = summe2 + L(i,k)*conj(L(j,k));
       end

   end
end
```

Dabei steht der Befehl `conj(L(j,k))` für das konjugiert komplexe von $l_{j,k}$. Nun muss nur noch das Element $l_{i,j}$ mit Formel (3) explizit berechnet werden:

```
for j=1:n
   summe1 = 0;
   for k=1:j-1
       summe1 = summe1 + (L(j,k))^2;
   end
   L(j,j)=sqrt(A(j,j)-summe1);
   for i = j+1:n
       summe2 = 0;
       for k=1:j-1
           summe2 = summe2 + L(i,k)*conj(L(j,k));
       end
       L(i,j)=(A(i,j)-summe2)/L(j,j);
   end
end
```

Damit wäre die Cholesky-Zerlegung programmiert! Nun fehlt noch das Lösen des Gleichungssystems mit Hilfe von Vorwärts- und Rückwärtseinsetzen. Dazu überlegen wir uns wieder erst, in welcher Reihenfolge welche Elemente berechnet werden. Zunächst soll das Gleichungssystem $Ly = b$ mittels Vorwärtseinsetzen gelöst werden, dabei ist L die linke untere Dreiecksmatrix, die wir mit der Cholesky-Zerlegung berechnet haben. Das Gleichungssystem hat also folgende Gestalt:

$$\begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ l_{2,1} & l_{2,2} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ l_{n,1} & l_{n,2} & \cdots & l_{n,n} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Daraus folgt

$$\begin{aligned} y_1 &= \frac{b_1}{l_{1,1}} \\ y_2 &= \frac{1}{l_{2,2}} (b_2 - l_{2,1} y_1) \\ y_3 &= \frac{1}{l_{3,3}} (b_3 - l_{3,1} y_1 - l_{3,2} y_2) \\ &\vdots \\ y_i &= \frac{1}{l_{i,i}} \left(b_i - \sum_{j=1}^{i-1} l_{i,j} y_j \right), i = 2, \ldots, n \end{aligned}$$

Nachdem wir den Vektor y mit `y=zeros(n,1)` initialisiert haben, berechnen wir das erste Element y_1 direkt. Für die Elemente y_i, $i = 2, \ldots, n$ führen wir eine Schleife ein.

```
y = zeros(n,1);
y(1) = b(1)/L(1,1);
for i=2:n
    summe = 0;
    for j=1:i-1
        summe = summe + L(i,j)*y(j);
    end
    y(i)=(b(i) - summe)/L(i,i);
end
```

Damit ist der Vektor y berechnet. Durch Rückwärtseinsetzen wird nun das Gleichungssystem $L^* x = b$ gelöst, dabei ist L^* eine rechte obere Dreiecks-

matrix, die komplex konjugierte von L.

$$\begin{pmatrix} l_{11} & \bar{l}_{21} & \cdots & \bar{l}_{n1} \\ 0 & l_{22} & \cdots & \bar{l}_{n2} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & l_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

Hier wird zuerst die letzte Zeile ausgewertet, es folgt:

$$\begin{aligned} x_n &= \frac{y_n}{l_{n,n}} \\ x_{n-1} &= \frac{1}{l_{n-1,n-1}} \left(y_{n-1} - \bar{l}_{n-1,n} x_n \right) \\ &\vdots \\ x_i &= \frac{1}{l_{i,i}} \left(y_i - \sum_{j=i+1}^{n} \bar{l}_{i,j} x_j \right) \end{aligned}$$

Nach der letzte Zeile wird die vorletzte ausgewertet usw. Also muss im Code eine Rückwärtsschleife eingebaut werden. Der Wert von x_n wird vorab berechnet, dann wird rückwärts bis x_1 gerechnet. Die Summe wird wieder wie oben implementiert. Für den Code ergibt sich daher

```
x = zeros(n,1);
R = L';
x(1) = y(1)/R(1,1);
for i=n-1:-1:1
    summe = 0;
    for j=i+1:n
        summe = summe + R(i,j)*x(j);
    end
    x(i)=(y(i) - summe)/R(i,i);
end
```

Wir schreiben die Cholesky-Zerlegung in ein Hauptprogramm `cholesky.m`, die Routinen zum Vorwärts- und Rückwärtseinsetzen werden in Funktionen `vorw.m` und `rueckw.m` gespeichert. Die einzelnen Programme sehen dann wie folgt aus.

- Hauptprogramm `cholesky.m`:

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%  cholesky    - Implementation des Cholesky-Algorithmus
%              fuer positiv definite Matrizen A fuer das
%              Gleichungssystem A x = b.
%  A           Matrix
%  b           Rechte Seite
%  x           Loesungsvektor
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

disp('Implementation des Cholesky-Algorithmus fuer ')
disp('positiv definite Matrizen A fuer das LGS A x = b. ')
disp(' ')
disp(' ')
disp('Bitte quadratische, symmetrische, ')
A = input('positiv definite Matrix A eingeben. A = ');
disp(' ')
disp(' ')
b = input('Bitte Loesungsvektor b eingeben. b = ');
disp(' ')
disp(' ')

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%  Dimension von A:
[n,m] = size(A);
if n~=m
     error('Matrix A ist nicht quadratisch !');
end

% Ist die Matrix A symmetrisch ?
if A == A'

else
    error('Matrix A ist nicht symmetrisch !');
end

% Ist die Matrix A positiv definit ?
% Anm.: Eigenwerte(A) > 0 und A symmetrisch
% => A positiv definit
% eig(A) ist ein Vektor der alle EW von A enthaelt.
% Ist einer davon <0, so ist die Matrix nicht
% pos. definit. Also reicht es, den kleinsten zu
```

```
% untersuchen.
if min(eig(A)) < 1.e-15
    error('Matrix A ist nicht positiv definit !');
end

% Ist b ein Spaltenvektor ?
[q,p] = size(b);
if q~=n
     error('b hat die falsche Dimension!');
end


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


%  Cholesky-Zerlegung A = L L'
%  L ist untere Dreiecksmatrix

L=zeros(n,n);

for j=1:n
    summe1 = 0;
    for k=1:j-1
        summe1 = summe1 + (L(j,k))^2;
    end
    L(j,j) = sqrt(A(j,j) - summe1);
    for i=j+1:n
        summe2 = 0;
        for k=1:j-1
            summe2 = summe2 + L(i,k) * conj(L(j,k));
        end
        L(i,j) = (A(i,j) - summe2) / L(j,j);
    end
end

% Ausgabe der gewonnenen Matrix L
disp ('Die untere Dreiecksmatrix L:');
disp(L);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Loesen des Gleichungssystems
% Schritt 1: Vorwaertseinsetzen, loese Ly=b
y = vorw(L,b,n);
```

```
% Schritt 2: Rueckwaertssubstitution Rx=y mit R=L'
x = rueckw(L',y,n);

disp('Loesungsvektor x=');
disp(x);
```

- Funktion `vorw.m`

```
function y = vorw(L,b,n)
% Die Funktion vorw.m loest das LGS Ly=b, L linke untere
% Dreiecksmatrix, durch Vorwaertseinsetzen

y = zeros(n,1);
y(1) = b(1)/L(1,1);
for i=2:n
    summe = 0;
    for j=1:i-1
        summe = summe + L(i,j)*y(j);
    end
    y(i)=(b(i) - summe)/L(i,i);
end
```

- Funktion `rueckw.m`

```
function x = rueckw(R,b,n)
% Die Funktion rueckw.m loest das LGS L'x=y, L'=R rechte
% obere Dreiecksmatrix, durch Rueckwaertseinsetzen

x = zeros(n,1);
x(n) = y(n)/R(n,n);
for i=n-1:-1:1
    summe = 0;
    for j=i+1:n
        summe = summe + R(i,j)*x(j);
    end
    x(i)=(y(i) - summe)/R(i,i);
end
```

Zu beachten ist hier, dass die Funktion `rueckw.m` mit der Matrix `L'`$=L^*$ aufgerufen wird (`x = rueckw(L',y,n);`), in der Funktion `rueckw.m` selbst wird L^* aber R genannt (`function x = rueckw(R,b,n)`).