Numerische Lineare Algebra

Proseminar

Cholesky-Zerlegung

von

Franz Brauße

vorgelegt am Fachbereich IV
der Universität Trier

bei

Frau Dipl.-Math. Christina Jager

06.02.2012

Inhaltsverzeichnis

1 Einleitung

Es werden in dieser Arbeit mehrere Zerlegungen von Matrizen A, die unterschiedliche Eigenschaften aufweisen, betrachtet. Im Allgemeinen kann A über dem Raum $\mathbb{C}$ betrachtet werden, um weniger Notation einführen zu müssen, wird jedoch nur der Raum $\mathbb{R}$ der reellen Zahlen zu Grunde gelegt.

Aus der LU-Zerlegung lässt sich durch wenige Umformungen die $LDM^\top$-Zerlegung gewinnen, die dann bei symmetrischen Matrizen besondere Eigenschaften aufweist und zur $LDL^\top$-Zerlegung wird. Es wird ein Algorithmus vorgestellt, der diese Zerlegung direkt berechnet.

Davon ausgehend wird die Cholesky-Zerlegung hergeleitet und es werden zwei Algorithmen zu deren Berechnung angegeben und ihre Stabilität kurz betrachtet. Es wird auch darauf eingangen, wie der Algorithmus für allgemeinere Matrizen angepasst werden muss.

Schließlich wird eine Variante der Pivotierung zur Erhöhung der Stabilität der Verfahren beschrieben, die die Symmetrie einer Matrix nicht zerstört, da diese notwendigerweise für die Verfahren erhalten bleiben muss.

2 Zerlegung für quadratische Matrizen

2.1 Herleitung der $LDM^\top$-Zerlegung

Aus der LU-Zerlegung mittels des Gaußschen Eliminationsverfahrens für beliebige Matrizen $A \in \mathbb{R}^{n\times m}$ in eine linke untere Dreieckmatrix L (im Folgenden abkürzend $L = \triangle$) und eine rechte obere Dreiecksmatrix U lässt sich für quadratische Matrizen eine weitere Zerlegung definieren.

Sei im Folgenden $A \in \mathbb{R}^{n\times n}$ quadratisch. Ist A nichtsingulär, so gibt es untere Dreicksmatrizen $L, M \in \mathbb{R}^{n\times n}$ und eine Diagonalmatrix $D = \operatorname{diag}(d_1, \dots, d_n)$, sodass $A = LDM^\top$ gilt. Diese Zerlegung ist eindeutig.

Ist die LU-Zerlegung von A bekannt, so kann daraus die $LDM^\top$-Zerlegung gewonnen werden. Setze

$$d_i = u_{ii}, \qquad i = 1, \dots, n$$

Da A nichtsingulär, gilt $\operatorname{rg}(U) = \operatorname{rg}(A) = n$ und es folgt, dass $D = \operatorname{diag}(d_i)_i$ dann ebenfalls regulär ist.

Also existiert $D^{-1} = \operatorname{diag}\left(\frac{1}{d_i}\right)_i$, womit sich schließlich $M^\top$ ergibt zu

$$M^\top = D^{-1}U$$

L wird unverändert aus der LU-Zerlegung übernommen.

Zusammengesetzt ergibt sich

$$LDM^\top = LD(D^{-1}U) = LU = A.$$

Die Eindeutigkeit der $LDM^\top$-Zerlegung folgt direkt aus der der LU-Zerlegung.

Seien $L'D'M'^\top = A = LDM^\top$ zwei Zerlegungen von A und $\overline{LU} = A$ ihre LU-Zerlegung. Dann

$$\begin{aligned} d'_i = \overline{u}_{ii} = d_i \quad &\Longrightarrow \quad D' = D \\ &\Longrightarrow \quad D'^{-1} = D^{-1} \\ &\Longrightarrow \quad M' = D'^{-1}\overline{U} = D^{-1}\overline{U} = M \\ \text{und nach Def.} \qquad &\qquad L' = \overline{L} = L. \end{aligned}$$

Damit ist die $LDM^\top$-Zerlegung lediglich eine andere Art, $A = LU$ auszudrücken. Wie in Abschnitt 2.4 beschrieben wird, hat diese Zerlegung jedoch Eigenschaften, die sich bei bestimmten Matrizen besser ausnutzen lassen.

2.2 Direkte Berechnung der $LDM^\top$-Zerlegung

Statt von einer vorhandenen LU-Zerlegung von A auszugehen, kann $A = LDM^\top$ auch direkt berechnet werden. Die Herleitung des Algorithmus geht in Schritten induktiv vor.

Berechne in jedem Schritt eine Spalte von L, eine Zeile von $M^\top$ und einen Diagonalwert von D.

Sind die Teilmatrizen $L^{(j)} = (l_{*k})_{k<j}$ und $M^{(j)} = (m_{i*})_{i<j}$, sowie die Diagonalwerte $d_1, \ldots, d_{j-1}$ von D im Schritt j bekannt, dann lässt sich daraus $L^{(j+1)}, M^{(j+1)}$ sowie d_j gewinnen:

Betrachte den Vektor v der j-ten Spalte der Gleichung $A = LDM^\top$:

$$(1) \qquad a_{*j} = Lv \qquad \text{und} \qquad v = DM^\top e_j \qquad (2)$$

Die „obere“ Hälfte von (1) weist je v_i $(i = 1, \ldots, j)$ als Lösung des linearen Gleichungssystems mit der bekannten unteren Dreiecksmatrix $L^{(j)}$ aus, d.h.

$$\sum_{k=1}^{n} l_{ik} v_k = \sum_{k=1}^{j} l_{ik} v_k = a_{ij}, \qquad i = 1, \ldots, j.$$

Für $i = 1, \ldots, j$ ist v_i damit berechnet. Dies in $v = DM^\top e_j$ eingesetzt ergibt dann

$$\begin{aligned} v_i &= \left(\sum_{k=1}^{n} d_{ik} m_{kl}^\top \right)_{l=1}^{n} \cdot e_j \\ &= \sum_{k=1}^{n} d_{ik} m_{kj}^\top \\ &= d_i m_{ij}^\top, \qquad i = 1, \ldots, j \end{aligned}$$

Damit lässt sich d_j und $M^{(j+1)}$ bestimmen durch

$$\begin{aligned} d_j &= v_j, \\ m_{ji} &= \frac{v_i}{d_i}, \qquad i = 1, \ldots, j-1. \end{aligned}$$

Nach Definition $M = D^{-1}U$ gilt $m_{jj} = 1$ für alle j.
Die j-te Spalte von L lässt sich nun mit der unteren Hälfte von (1),

$$a_{ij} = L_{i*}^{(j)} v,$$

errechnen.
Für $i = j+1, \ldots, n$ gilt

$$\begin{aligned} & & a_{ij} &= \sum_{k=1}^{j} l_{ik} v_k \\ \iff & & l_{ij} v_j &= a_{ij} - \sum_{k=1}^{j-1} l_{ik} v_k. \end{aligned}$$

Beispiel

Die $LDM^\top$-Zerlegung von $A = \begin{bmatrix} 1 & 3 \\ 4 & 2 \end{bmatrix}$ errechnet sich nach obigem Algorithmus also wie folgt.

$j = 1$: $L = \begin{bmatrix} 1 & 0 \\ \cdot & 1 \end{bmatrix}$, $D = \begin{bmatrix} \cdot & 0 \\ 0 & \cdot \end{bmatrix}$, $M^\top = \begin{bmatrix} 1 & \cdot \\ 0 & 1 \end{bmatrix}$

$$\begin{aligned} a_{11} &= \sum_{k=1}^{1} l_{1k} v_k = l_{11} v_1 = v_1 \qquad \Longrightarrow \quad v_1 = 1 = d_1 \\ l_{21} &= \frac{a_{21} - \sum_{k=1}^{0} l_{ik} v_k}{v_1} = \frac{a_{21}}{1} = 4 \end{aligned}$$

$j = 2$: $L = \begin{bmatrix} 1 & 0 \\ 4 & 1 \end{bmatrix}$, $D = \begin{bmatrix} 1 & 0 \\ 0 & \cdot \end{bmatrix}$, $M^\top = \begin{bmatrix} 1 & \cdot \\ 0 & 1 \end{bmatrix}$

$$\begin{aligned} a_{12} &= \sum_{k=1}^{2} l_{1k} v_k = v_1 + 0 \qquad \Longrightarrow \quad v_1 = 3 \\ a_{22} &= \sum_{k=1}^{2} l_{2k} v_k = 4 \cdot v_1 + v_2 \quad \Longrightarrow \quad v_2 = -10 = d_2 \\ m_{21} &= \frac{v_1}{d_1} = \frac{3}{1} = 3 \end{aligned}$$

Damit ergibt sich

$$L = \begin{bmatrix} 1 & 0 \\ 4 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 1 & 0 \\ 0 & -10 \end{bmatrix}, \quad M^\top = \begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix}.$$

2.3 Algorithmische Umsetzung

Da die Spalte j von A nach Schritt j nicht mehr benötigt wird, können – analog zur LU-Zerlegung – dort die berechneten l_{*j}, d_j sowie $m_{*j}^\top$ abgespeichert werden, wie in Abbildung 1 angedeutet.

$$\begin{bmatrix} d_1 & m_{12}^\top & \cdots & m_{1j}^\top & a_{1,j+1} & \cdots & a_{1n} \\ l_{21} & d_2 & \ddots & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \ddots & m_{j-1,j}^\top & \vdots & & \vdots \\ \vdots & & \ddots & d_j & \vdots & & \vdots \\ \vdots & & & l_{j+1,j} & \vdots & & \vdots \\ \vdots & & & \vdots & \vdots & & \vdots \\ l_{n1} & \cdots & \cdots & l_{nj} & a_{n,j+1} & \cdots & a_{nn} \end{bmatrix}$$

Abbildung 1: Darstellung der Speicherstruktur nach Schritt j, in welcher der Algorithmus die Matrix A durch ihre $LDM^\top$-Zerlegte ersetzt.

1.	v_i	$=$	Lösung von $L_{i*}^{(j)} v = a_{ij}$,	$i = 1, \ldots, j$
2.	d_j	$=$	v_j	
3.	m_{ji}	$=$	$\frac{v_i}{d_i}$,	$i = 1, \ldots, j$
4.	l_{ij}	$=$	$\frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} v_k}{v_j}$,	$i = j+1, \ldots, n$

Abbildung 2: Algorithmus zur Berechnung der Zerlegung $A = LDM^\top$ in jedem Schritt $j = 1, \ldots, n$.

Laufzeitabschätzung

Pro Schritt $j = 1, \ldots, n$ ist folgender Rechenaufwand zu erwarten.

1. Je j-fache Rücksubstitution $\implies \sum_{j=1}^{n} \sum_{k=1}^{j} 2k$ flops $\in \mathcal{O}\left(\frac{n^3}{3}\right)$.
2. Je eine Zuweisung $\implies \sum_{j=1}^{n} 1 = n$ flops.

3. Je j-fache Division $\implies \sum_{j=1}^{n} j$ flops $\in \mathcal{O}\left(\frac{n^2}{2}\right)$.

4. Je $(n-j)$-faches, j-maliges Aufsummieren und eine Division

$$\implies \sum_{j=1}^{n}(n-j)\cdot 2j = n^2(n+1) - \frac{n(n+1)(2n+1)}{3} = \frac{n^3-n}{3} \text{ flops} \in \mathcal{O}\left(\frac{n^3}{3}\right).$$

Zusammengenommen ergibt sich also ein algorithmischer Aufwand von $\mathcal{O}(\frac{2}{3}n^3)$ Floating-Point-Operationen (flops).

2.4 Zerlegung symmetrischer Matrizen

Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und regulär. Dann gilt für ihre $LDM^\top$-Zerlegung $L = M$.

Betrachte $L = \llcorner$ und die Produkte $C = LAL^\top$, $B = LA$. Behauptung: C ist ebenfalls symmetrisch.

$$\begin{aligned}
c_{ij} &= \sum_f b_{if} l_{fj}^\top = \sum_f \left(\sum_k l_{ik} a_{kf}\right) l_{jf} \\
&= \sum_f \sum_k l_{ik} a_{kf} l_{jf} \overset{a_{kf}=a_{fk}}{=} \sum_k \sum_f l_{jf} a_{fk} l_{ik} \\
&= \sum_k \left(\sum_f l_{jf} a_{fk}\right) l_{ki}^\top = \sum_k b_{jk} l_{ki}^\top \\
&= c_{ji}
\end{aligned}$$

Also ist $C = LAL^\top$ ebenfalls symmetrisch, wenn A symmetrisch war.

Damit folgt

$$\begin{aligned}
& A = LDM^\top \text{ symmetrisch} \\
\implies \quad & M^{-1}AM^{-\top} = \underbrace{M^{-1}}_{\llcorner} \cdot \underbrace{LD}_{\llcorner} = \llcorner \text{ ebenfalls symmetrisch} \\
\implies \quad & M^{-1}LD \text{ ist Diagonalmatrix.}
\end{aligned}$$

Da $M^{-1}LD$ eine Diagonalmatrix und D regulär ist, muss auch $M^{-1}L$ eine Diagonalmatrix sein. Wegen $m_{ii} = l_{ii} = 1$ folgt $M^{-1}L = I$, damit gilt $M = L$.

Also existieren eine linke untere Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ und eine Diagonalmatrix $D = \operatorname{diag}(d_1, \ldots, d_n) \in \mathbb{R}^{n\times n}$ mit $A = LDL^\top$.

Ist im Voraus bekannt, dass A symmetrisch ist, können beim Algorithmus zur Berechnung dieser Zerlegung sowohl Speicherplatz als auch Rechenoperationen eingespart werden, wie im Folgenden beschrieben wird.

Da im j-ten Schritt bereits l_{jk} für $k < j$ bekannt ist, entfällt das Lösen des $(j \times j)$-Gleichungssystems $Lv = A$ im Algorithmus zur $LDM^\top$-Zerlegung, da v auch durch $v = DM^\top e_j = DL^\top e_j$, d.h. ein bloßes Skalieren der bereits bekannten Werte l_{jk} durch d_k, berechnet werden kann:

$$v = \begin{pmatrix} d_1 \cdot l_{j1} \\ \vdots \\ d_{j-1} \cdot l_{j,j-1} \\ d(j) \end{pmatrix}$$

Es ergibt sich der in Abbildung 3 angegebene Algorithmus.

```
1 for j = 1 : n
2     for i = 1 : j − 1
3         v(i) = L(j, i)d(i)
4     end
5     v(j) = A(j, j) − L(j, 1 : j − 1)v(1 : j − 1)
6     d(j) = v(j)
7     L(j + 1 : n, j) = (A(j + 1 : n, j) − L(j + 1 : n, 1 : j − 1)v(1 : j − 1)) / v(j)
8 end
```

Abbildung 3: Algorithmus zur Berechnung der $LDL^\top$-Zerlegung von $A \in \mathbb{R}^{n \times n}$ symmetrisch.

Laufzeitabschätzung

- Die Laufzeit der Zeilen 2-4 und 5 ergeben wieder eine Arithmetische Reihe, d.h. $\mathcal{O}(n^2)$ flops.
- Zeile 6 hat lineare Laufzeit in n flops.
- An Zeile 7 hat sich im Vergleich zum vorherigen Algorithmus in Abb. 2 nichts verändert; ihre Laufzeit lässt sich weiterhin durch $\mathcal{O}\left(\frac{n^3}{3}\right)$ flops abschätzen.

Damit hat der Algorithmus eine geringere, im Wesentlichen halbierte, Laufzeit von nur noch $\mathcal{O}\left(\frac{n^3}{3}\right)$ flops, was auf die Symmetrie von A zurückzuführen ist.

3 Zerlegung positiv (semi-)definiter Matrizen

3.1 Positive Definitheit

Eine Matrix $A \in \mathbb{R}^{n \times n}$ heißt positiv definit, falls für alle $0 \neq x \in \mathbb{R}^n$ gilt

$$x^\top Ax > 0$$

Anwendungen positiv definiter Matrizen umfassen bspw.

- Quadriken $x^\top Ax + b^\top x = 1$: A bei Ellipsoiden hat nur positive Eigenwerte, ist deshalb auch pos. def.
- $X^\top X$ ist pos. def. falls X vollen Rang hat.
- Optimierungsprobleme, strikt konvexe Funktion mit einem globalen Minimum

Positive Definitheit wird eher selten ohne gleichzeitige Symmetrie betrachtet, obwohl es solche Matrizen gibt, bspw. $\left[\begin{smallmatrix} 1 & 1 \\ -1 & 1 \end{smallmatrix}\right]$.

3.2 Rundungsfehler und Auslöschung

Arithmetische Operationen $x \circ y$ für $\circ \in \{+, -, \cdot, /\}$, aber beispielsweise auch $\sqrt[2]{x}$, $\operatorname{sincos}(x)$, $2^x - 1$, $y \cdot \log_2 x$ für $x, y \in \mathbb{R}$, sowie $x^\top y$ für $x, y \in \mathbb{R}^n, n \leq 8$ können als Maschinenbefehle (1 flop) ausgeführt werden, wenn man mit einer Approximation[1] der *Maschinengenauigkeit* $\mathbf{u}$ der Zahlen arbeitet[2]. Es gibt zwei standardisierte und gängige Datentypen, die hardwareseitig unterstützt werden:

- Single (`float`, $\mathbf{u} = \frac{2^{-23}}{2} \approx 10^{-7}$) bzw.
- Double Precision (`double`, $\mathbf{u} = \frac{2^{-52}}{2} \approx 10^{-16}$).

Selbstverständlich ist es möglich, eigene Datentypen und Operationen auf ihnen zu definieren, die eine höhere Genauigkeit bieten. Aufgrund des begrenzten Speicherplatzes ist es jedoch im Allgemeinen unmöglich, beliebige reelle Zahlen zu speichern. Ein weiteres Problem ist die Geschwindigkeit, mit der diese Operationen durchgeführt werden. Obige Operationen können innerhalb weniger Taktzyklen direkt auf modernen Prozessoren ausgeführt werden, während bei eigenen Datentypen in der Regel keine Hardwareunterstützung vorhanden und auch ihr Rundungsverhalten nicht festgelegt ist, im Gegensatz zu **`float`** und **`double`**.

Rundungsfehler treten bei den standardisierten Datentypen vor allem bei der Verrechnung von großen mit kleinen Zahlen auf.

Beispiel zu Auslöschung bei 8-stelliger Dezimalarithmetik:

$$\begin{array}{rcl} 0{,}12345679 & \leftarrow & \text{relativer Fehler} \approx 10^{-8} \text{ in Eingabe} \\ -0{,}12345678 & & \\ \hline 0{,}00000001 & \leftarrow & \text{relativer Fehler} \approx 1 \text{ in Ausgabe} \end{array}$$

3.2.1 Stabilität von $LDM^\top$ für positiv definite Matrizen

Sei $\varepsilon > 0$, dann ist A positiv definit:

$$A = \begin{bmatrix} \varepsilon & m \\ -m & \varepsilon \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{m}{\varepsilon} & 1 \end{bmatrix} \begin{bmatrix} \varepsilon & 0 \\ 0 & \varepsilon + \frac{m^2}{\varepsilon} \end{bmatrix} \begin{bmatrix} 1 & \frac{m}{\varepsilon} \\ 0 & 1 \end{bmatrix} = LDM^\top$$

[1] Standard IEEE-754-2008 beschreibt deren Aufbau und Rundungsverhalten und wird von allen gängigen Prozessoren, die Gleitkommaarithmetik beherrschen, unterstützt.

[2] Damit liegen die betrachteten Zahlen x, y dann eigentlich nicht mehr in $\mathbb{R}$, sondern nur noch in einem Teilraum von $\mathbb{Q}$.

Falls $\frac{m}{\varepsilon} \gg 1$, so wird die vorgestellte $LDM^\top$-Zerlegung instabil und es kommt zu stärkeren Rundungsfehlern und damit verbunden Auslöschungen.

Betrachtet man die Anteile $T = \frac{1}{2}(A + A^\top)$ der Symmetrie und $S = \frac{1}{2}(A - A^\top)$ die der Asymmetrie, dann lässt[3] sich die absolute Größe der Einträge in $LDM^\top = A$ abschätzen durch

$$\| |L| \cdot \underbrace{|D|}_{(|d_{ij}|)_{ij}} \cdot |M^\top| \|_F \leq n(\|T\|_2 + \|ST^{-1}S\|_2).$$

Wurde $\hat{x}$ berechnet als Lösung zu $Ax = b$, dann beschreibt dieses Rundungsfehlerbehaftete $\hat{x}$ die Lösung des um $E \in \mathbb{R}^{n \times n}$ gestörten Systems $(A + E)\hat{x} = b$.

E lässt sich nun abschätzen durch (c abhängig von $\| |\hat{L}||\hat{D}||\hat{M}^\top| \|_F$)

$$\|E\|_F \in \mathcal{O}(n\|A\|_F + \mathbf{u}cn^2(\|A\|_2 + \|ST^{-1}S\|_2) + \mathbf{u}^2).$$

Golub und van Loan kommen zu dem Schluss, dass, falls

$$\Omega = \frac{\|ST^{-1}S\|_2}{\|A\|_2}$$

„nicht zu groß“ ist, auch nicht pivotiert werden muss. Das heißt die Norm des asymmetrischen Anteils S darf im Vergleich zu dem des symmetrischen Teils T nicht zu groß werden, um die $LDM^\top$-Zerlegung wie beschrieben ohne Pivotisierung stabil durchführen zu können.

Ist A symmetrisch, dann ist $S = 0$ und damit ebenfalls $\Omega = 0$. Wegen dieser Eigenschaft und weil der Fokus dieser Arbeit auf symmetrischen Matrizen liegt, die in den folgenden Abschnitten behandelt werden, werden keine Beweise der obigen Aussagen gegeben.

3.3 Symmetrisch positiv definite Matrizen

Eine Eigenschaft symmetrisch positiver Matrizen ist, dass ihre Elemente auf der Hauptdiagonalen alle anderen dominieren (ohne Beweis).

Für A sym. pos. def. existiert $LDL^\top = A$ und ist stabil zu berechnen.

L ist eine strikte untere Dreiecksmatrix, d.h. für ihre Einträge auf der Hauptdiagonalen gilt $l_{ii} = 1$.

Die Diagonalmatrix D enthält wegen A pos. def. nur positive Einträge auf der Diagonalen.

Deshalb lässt sich D nun in die beiden Faktoren L bzw. $L^\top$ integrieren, sodass eine neue Zerlegung von A entsteht.

3.4 Cholesky-Zerlegung

Sei $G = L\,\mathrm{diag}(\sqrt{d_1}, \ldots, \sqrt{d_n})$, dann $\mathbb{R}^{n \times n} \ni G = \triangle$ und es gilt:

$$A = GG^\top.$$

[3] Beweis siehe Golub, Van Loan: „Unsymmetric Positive Definite Linear Systems“ (1979).

G wird auch Cholesky-Dreieck genannt.

Offenbar

$$\begin{aligned} GG^\top &= (L\,\mathrm{diag}(\sqrt{d_i})_i)(\mathrm{diag}(\sqrt{d_i})_i L^\top) \\ &= L\,\mathrm{diag}(d_i)_i L^\top = LDL^\top. \end{aligned}$$

Das System $Ax = b$ lässt sich nun wieder durch zweifache Rücksubstitution $Gy = b$ und $G^\top x = y$ lösen. Dann

$$b = Gy = G(G^\top x) = (GG^\top)x = Ax.$$

Ein Vorteil ist, dass nur eine Dreicksmatrix G abgespeichert werden muss.

Beispiel

Die Matrix $A = \begin{bmatrix} 2 & -2 \\ -2 & 5 \end{bmatrix}$ ist symmetrisch und positiv definit.

Ihre $LDL^\top$-Zerlegung erfolgt in zwei Schritten:

$j = 1$:

$$\begin{aligned} v_1 &= a_{11} - 0 = 2 \\ d_1 &= v_1 = 2 \\ l_{21} &= \frac{a_{21} - 0}{v_1} = -1 \end{aligned}$$

$j = 2$:

$$\begin{aligned} v_1 &= l_{21} d_1 = -2 \\ v_2 &= a_{22} - l_{21} v_1 = 3 \\ d_2 &= v_2 = 3 \end{aligned}$$

Damit ergibt sich

$$A = \underbrace{\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}}_{L^\top} = \underbrace{\begin{bmatrix} \sqrt{2} & 0 \\ -\sqrt{2} & \sqrt{3} \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} \sqrt{2} & -\sqrt{2} \\ 0 & \sqrt{3} \end{bmatrix}}_{G^\top}$$

3.5 Effiziente Algorithmen

3.5.1 Äußeres-Produkt-Updates

Der Algorithmus wird $A \in \mathbb{R}^{n \times n}$ zerlegen in ein Skalar $\beta \in \mathbb{R}$, einen Vektor $v \in \mathbb{R}^{n-1}$ und eine neue Matrix $A' \in \mathbb{R}^{(n-1)\times(n-1)}$, mit der diese Prozedur dann wiederholt wird.

Betrachte die Partitionierung

$$\begin{aligned} A = \begin{bmatrix} \alpha & v^\top \\ v & B \end{bmatrix} &= \begin{bmatrix} \beta & 0 \\ \frac{v}{\beta} & I_{n-1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & B - \frac{1}{\alpha} v v^\top \end{bmatrix} \begin{bmatrix} \beta & \frac{v^\top}{\beta} \\ 0 & I_{n-1} \end{bmatrix} \\ &= G_1 \begin{bmatrix} 1 & 0 \\ 0 & A' \end{bmatrix} G_1^\top. \end{aligned}$$

Dann ist $\beta = \sqrt{\alpha}$ und wegen A positiv definit gilt $\alpha > 0$, das heißt $\sqrt{\alpha}$ existiert. Die Ausgangsmatrix A' für die nächste Iteration ergibt sich aus der Teilmatrix B aktualisiert durch das skalierte äußere Produkt $-\frac{1}{\alpha} v v^\top$.

Soll diese Vorschrift induktiv fortgesetzt werden, so bleibt zu zeigen, dass A' immernoch positiv definit ist.

Dazu wird zunächst allgemeiner gezeigt, dass für $A \in \mathbb{R}^{n \times n}$ positiv definit und $X \in \mathbb{R}^{n \times k}$ mit Rang k gilt, dass

$$B = X^\top A X \in \mathbb{R}^{k \times k}$$

positiv definit ist.

Falls für $z \in \mathbb{R}^k$ gilt

$$0 \geq z^\top B z = (Xz)^\top A (Xz),$$

dann folgt $Xz = 0$, da A positiv definit ist.

Da X aber vollen Spaltenrang k hat, ist sein Kern $\{0\}$, d.h.

$$Xz = 0 \iff z = 0.$$

Also gilt $z^\top B z \leq 0 \iff z = 0$, womit die neue Matrix B ebenfalls positiv definit ist. $\square$

Sei nun $X = \begin{bmatrix} \frac{1}{\beta} & -\frac{v^\top}{\beta^2} \\ 0 & I_{n-1} \end{bmatrix} \in \mathbb{R}^{n \times n}$. Für $X^\top A X$ gilt mit Obigem positive Definitheit. Andererseits ist

$$X^\top G_1 = \begin{bmatrix} \frac{1}{\beta} & 0 \\ -\frac{v}{\beta^2} & I_{n-1} \end{bmatrix} \begin{bmatrix} \beta & 0 \\ \frac{v}{\beta} & I_{n-1} \end{bmatrix} = I,$$

d.h. $B - \frac{1}{\alpha} v v^\top$ ist eine Teilmatrix von $X^\top A X = X^\top G_1 A' G_1^\top X = A'$ und damit auch positiv definit.

Also sind alle Voraussetzungen für die iterierte Durchführung der Partitionierung erfüllt und es lässt sich nun G_2 und ein A'' mit dem selben Verfahren aus A' erzeugen. Damit ergibt sich

$$A = G_1 G_2 \dots G_{n-1} G_n \cdot I_n \cdot G_n^\top G_{n-1}^\top \dots G_2^\top G_1^\top$$

und der in Abbildung 4 angegebene Algorithmus. Zeile 5 führt die Anpassung von B mittels des äußeren Produkts $v v^\top$ durch.

Die Matrix $G = G_1 G_2 \dots G_n$ wird an alten Positionen im unteren Dreieck von A gespeichert, wo sich je die Vektoren v im Schritt k befanden.

Als Laufzeit ergibt sich wieder $\mathcal{O}(\frac{n^3}{3})$ flops wegen der Neuberechnung von $B \in \mathbb{R}^{n-k \times n-k}$ in jedem Schritt $1 \leq k \leq n$ in Zeilen 4-6.

```
1 for k = 1 : n
2     A(k, k) = √A(k, k)                                  /* β  */
3     A(k + 1 : n, k) = A(k + 1 : n, k)/A(k, k)           /* v/β  */
4     for j = k + 1 : n
5         A(j : n, j) = A(j : n, j) − A(j : n, k)A(j, k)  /* B_{*j} − vv_j^⊤/β  */
6     end
7 end
```

Abbildung 4: Cholesky-Zerlegung mittels außeres-Produkt-Updates.

```
1 for j = 1 : n
2     if j > 1
3         A(j : n, j) = A(j : n, j) − A(j : n, 1 : j − 1)A(j, 1 : j − 1)^⊤  /* GAXPY */
4     end
5     A(j : n, j) = A(j : n, j) / √A(j, j)  /* scaled GAXPY operation */
6 end
```

Abbildung 5: Cholesky-Zerlegung mittels GAXPY-Operationen.

3.5.2 GAXPY

Wie in Abschnitt 3.2 dargelegt, gibt es Maschineninstruktionen auf neueren Prozessoren[4], die $x^\top y$ für $x, y \in \mathbb{R}^n$, $n \leq 8$ berechnen können. In der Numerik hat sich eine anspruchsvollere Operation herausgebildet, auf deren Benutzung numerische Algorithmen hin optimiert werden, die jedoch auch mit diesen einfacheren Operationen deutlich beschleunigt werden kann.

Die Operation heißt in ihrer auf Matrizen verallgemeinerten Variante *GAXPY*[5] und bezeichnet die Berechnung von $Ax + y$ für $A \in \mathbb{R}^{n\times m}$, $x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$.

Eine Fokussierung der Optimierung der Algorithmen auf wenige Grundoperationen ist hilfreich wenn anzunehmen ist, dass diese Grundoperationen schnell umgesetzt werden können – wie im Fall von GAXPY.

Eine Variante des Algorithmus zur Cholesky-Zerlegung, die Gebrauch von GAXPY-Operationen macht, ist in Abbildung 5 dargestellt.

Dieser Algorithmus benötigt nur halb so viele Vektoroperationen wie der Obige, hat allerdings auch eine Laufzeit von $\mathcal{O}(\frac{n^3}{3})$.

[4] bspw. solche mit AVX-Instruktionssatzerweiterung

[5] Generalized SAXPY, wobei letzteres für die Operation $ax + y$ auf Skalaren steht.

3.5.3 Eigenschaften der Cholesky-Zerlegung

Wie in der Herleitung bereits gesehen, folgt aus der Symmetrie und positiven Definitheit von A, dass ihre Cholesky-Zerlegung existiert und der Algorithmus immer positive Werte α auf der Hauptdiagonalen findet, sodass er dessen Wurzel berechnen kann.

Es gilt jedoch auch, dass falls der Algorithmus einen nicht-positiven Wert α vorfindet, dass im Umkehrschluss die Matrix A nicht positiv definit gewesen sein kann. Der Algorithmus angewandt auf eine symmetrische Matrix A kann also auch benutzt werden, um positive Definitheit festzustellen.

Zur Stabilität:

Die Einträge in G sind begrenzt durch die Diagonaleinträge von A:

$$\|A\|_2 = \|GG^\top\|_2 = \|G\|_2^2 \implies g_{ij}^2 \leq \sum_{k=1}^{i} g_{ik}^2 = a_{ii}$$

Der Rundungsfehler einer Lösung $\hat{x}$ von $(A+E)\hat{x} = b$ wird begrenzt durch

$$\|E\|_2 \leq c_n \mathbf{u} \|A\|_2,$$

wobei c_n eine kleine, von n abhängige Konstante ist (siehe Wilkinson 1968).

3.6 Erweiterung für positiv semidefinite Matrizen

Der vorgestellte Algorithmus lässt sich sehr einfach auf positiv semidefinite Matrizen A erweitern, indem ein Test auf $\alpha > 0$ vor der eigentlichen Berechnung innerhalb der Iteration eingefügt wird und andernfalls diese Zeile bzw. Spalte ignoriert wird.

Eine Begründung wird im Folgenden gegeben.

Nach Definition ist A positiv semidefinit, falls $\forall x \in \mathbb{R}^n : x^\top Ax \geq 0$.

Einige Eigenschaften positiv semidefiniter Matrizen $A = (a_{ij})_{ij}$:

1. $|a_{ij}| \leq (a_{ii} + a_{jj})/2$
2. $|a_{ij}| \leq \sqrt{a_{ii} a_{jj}}$
3. $\max\limits_{i,j} |a_{ij}| = \max\limits_{i} a_{ii}$
4. $a_{ii} = 0 \implies a_{i*} = a_{*i} = 0$

Denn:

1. $$\begin{aligned} x = e_i + e_j \quad &\implies \quad 0 \leq x^\top Ax = a_{ii} + a_{jj} + 2a_{ij} \quad \text{und} \\ x = e_i - e_j \quad &\implies \quad 0 \leq x^\top Ax = a_{ii} + a_{jj} - 2a_{ij}. \end{aligned}$$

2. Ohne Einschränkung seien $i = 1$ und $j = 2$ (da Teilmatrizen von positiv semidefiniten Matrizen mit gleicher Hauptdiagonalen ebenfalls positiv semidefinit sind).

```
1 for k = 1 : n
2     if A(k, k) > 0
3         A(k, k) = √A(k, k)
4         A(k + 1 : n, k) = A(k + 1 : n, k)/A(k, k)
5         for j = k + 1 : n
6             A(j : n, j) = A(j : n, j) − A(j : n, k)A(j, k)
7         end
8     end
9 end
```

Abbildung 6: Cholesky-ähnliche Zerlegung positiv semidefiniter Matrizen.

Da $A(1:2, 1:2)$ ebenfalls positiv semidefinit ist, folgt dann für alle $x \in \mathbb{R}$

$$0 \leq \begin{pmatrix} x \\ 1 \end{pmatrix}^\top \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} = a_{11}x^2 + 2a_{12}x + a_{22}$$

$\implies$ Diskriminante $4a_{12}^2 - 4a_{11}a_{22}$ der quadratischen Gleichung muss negativ sein.

3. bzw. 4. folgen aus 1. bzw. 2. □

Nun lässt sich eine Cholesky-ähnliche Operation auch auf symmetrisch positiv semidefiniten Matrizen anwenden.

Angenommen $a_{kk} = 0$, dann folgt mit 4., dass die gesamte Zeile k und Spalte k leer sind, das heißt $a_{ik} = a_{ki} = 0$ $(k \leq i \leq n)$ gilt, und im Algorithmus für G_k nichts zu tun bleibt.

Es ergibt sich durch einfache Abfrage auf $a_{kk} > 0$ der in Abbildung 6 aus in Abbildung 4 vorgestellten.

Rundungsfehler und Auslöschungserscheinungen, wie im Beispiel in Abschnitt 3.2 beschrieben, können bei der Abfrage jedoch fehlerhafte „wahr“-Instanzen liefern, die im Folgenden dann in Zeile 4 zu deutlich falsche Ergebnissen führen.

Deshalb wird in einer rundungsfehlerbehafteten Implementierung typischerweise nicht $a_{kk} > 0$ sondern $a_{kk} > \varepsilon$ für ein fixes ε, das im Wesentlichen von der Maschinengenauigkeit $\mathbf{u}$ und der Anzahl der Rechenschritte des Algorithmus abhängt, gewählt. Die Anzahl der Rechenschritte gibt einen Hinweis darauf, wie häufig Rundungsfehler das zu vergleichende Ergebnis a_{kk} in diesem Fall beinflussen. ε bleibt jedoch in aller Regel eine Schätzung, sodass diese krude Methode bei „bösartigen“ Eingabematrizen, die explizit Zahlen $a_{kk} \leq \varepsilon$ beinhalten oder hervorbringen, zu falschen Zerlegungen führt.

4 Symmetrische Pivotierung

Um solche sich stark auswirkende Rundungsfehler zu vermeiden wurde beispielsweise bei der *LU*-Zerlegung mit Pivotierung gearbeitet. Dazu wurden die Zeilen, die je den größten Wert in der Restmatrix beinhaltete an die Stelle permutiert, die der Algorithmus

als nächstes betrachtete. Formal ausgedrückt betrachtet der Algorithmus zur Zerlegung dann nicht mehr die ursprüngliche Eingabematrix A, sondern die vorher mit einer Permutationsmatrix P multiplizierte Matrix PA und zerlegt diese. P wird separat (optimiert) gespeichert und kann leicht aus dem Ergebnis entfernt werden:

$$PA = LU \iff P^{-1}PA = A = P^{-1}LU$$

Eine Invertierung von P kann leicht erechnet werden, denn es gilt $P^{-1} = P^\top$.

Sei $P \in \mathbb{R}^{n \times n}$ Permutationsmatrix, dann hat PA dieselben Zeilen wie A, jedoch anders sortiert, also bleibt die Symmetrie von A dabei nicht erhalten.

Eine die Symmetrie erhaltende Permutation wird durch $PAP^\top$ erreicht, bei der die Diagonalelemente auf der Diagonalen bleiben, jedoch anders angeordnet werden. P von links sortiert die Zeilen um, die Inverse $P^\top$ von rechts sortiert die Spalten um. Die Spalten j werden hierbei um genauso viele Einträge rotiert wie die entsprechenden Zeilen j.

Beispiel

P sei die zur Permutation ρ mit $(1, 2, 3, 4) \mapsto (2, 3, 4, 1)$ gehörende Permutationsmatrix.

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \\ a_{11} & a_{12} & a_{13} & a_{14} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} a_{22} & a_{23} & a_{24} & a_{21} \\ a_{32} & a_{33} & a_{34} & a_{31} \\ a_{42} & a_{43} & a_{44} & a_{41} \\ a_{12} & a_{13} & a_{14} & a_{11} \end{bmatrix}$$

Es ist erkennbar, dass die resultierende Matrix $A' = PAP^\top = (a'_{ij})_{ij}$ weiterhin symmetrisch ist und die Elemente a_{kk} auf der Hauptdiagonalen entsprechend ρ auf $a'_{\rho(k),\rho(k)}$ permutiert wurden.

Literatur

[1] Prof. Dr. Schulz, *Skript Numerik '08/'09*, Trier

[2] L.N. Trefethen, D. Bau, *Numerical Linear Algebra*

[3] G.H. Golub, C.T. van Loan, *Matrix computations*