Kapitel 2

Lineare Gleichungssysteme

Eine der wichtigsten numerischen Aufgaben ist das Lösen von Gleichungssystemen, denn dieses Problem taucht in vielen Verfahren als Teilproblem auf. Hierzu gibt es im wesentlichen zwei Typen von Algorithmen. Die direkten Verfahren, die in endlich vielen Schritten eine Lösung berechnen, die mit Rundungsfehlern und Ihren Konsequenzen behaftet ist. Andererseits die indirekten, iterativen Verfahren, die abbrechen, wenn eine „gewisse "Genauigkeit erreicht ist. Wir werden uns aus Zeitmangel nur mit direkten Verfahren beschäftigen.

Im gesamten Kapitel werden wir folgende Fragestellung untersuchen. Gegeben seien $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$. Gesucht ist ein Vektor $x \in \mathbb{R}^n$ mit $Ax = b$.

2.1 Gaußelimination und LU-Zerlegung, Pivotstrategien

Diese Aufgabenstellung ist bereits in der „Mathematik für Chemiker und Wirtschaftsinformatiker" behandelt worden. Wir wollen das dort vorgestellte Eliminationsverfahren etwas implementationsnäher darstellen und untersuchen.

Znächst einmal wiederholen wir die Vorgehensweise anhand eines Beispiels.

Beispiel 2.1.1 *Sei*

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 10 \\ -4 \\ -3 \\ -2 \end{pmatrix}.$$

$$
\begin{array}{cccc|c}
1 & 1 & 1 & 1 & 10 \\
-1 & 0 & 0 & 0 & -4 \\
0 & -1 & 0 & 0 & -3 \\
0 & 0 & -1 & 0 & -2
\end{array}
\qquad
\begin{array}{cccc|c}
\boxed{1} & 1 & 1 & 1 & 10 \\
0 & \boxed{1} & 1 & 1 & 6 \\
0 & 0 & \boxed{1} & 1 & 3 \\
0 & 0 & 0 & 1 & 1
\end{array},
$$

Und wir erhalten als Lösung $x_4 = 1, x_3 = 3-1 = 2, x_2 = 6-2-1 = 3, x_1 = 10-3-2-1 = 4$.

Vereinfacht formuliert geht man so vor, daß man von links nach rechts mit den Diagonalelementen eine *Gaußelimination* durchführt, d.h. man formt das System so äquivalent um, daß unter dem *Pivotelement* nur noch Nullen stehen. Notieren wir zunächst den *Gaußeliminationsschritt bzgl. des Elementes* a_{ij}.

```
def gausselim(A,i,j):
   for l in range(i+1,m):
      A_lj=A[l,j]
      A[l,:]=A[l,:]-A_lj/A[i,j]*A[i,:]
```

Bei der Bestimmung des Pivotelementes entlang der Diagonalen kann der Fall eintreten, daß $a_{jj} = 0$ ist. Gibt es dann ein $a_{kl} \neq 0$ mit $k \geq j$ und $l \geq j$, so vertauschen wir die k-te und j-te Zeile, sowie l-te und j-te Spalte, merken uns, daß in der Lösung die Indizes l und j vertauscht werden müssen und führen eine Gaußelimination mit a_{jj} durch.

```
def gaussalg(A,b):
  d=min([m,n])
  index=range(0,n)                # Permutation der Variablen
  C=concatenate((A,b),1)          # Verschmelze A,b nebeneinander
  for k in range(d):
    if C[k,k]==0:
      pivotfound,i,j = findpivot(C,index,k)
      if pivotfound == 0:
        break
      else:
        if i != k:                    # swap rows i and k
          swaprows(C,i,k)
        if j != k:                    # swap columns
          swapcolumns(C,index,j,k)    # and variables j and k
    gausselim(C,k,k)
  solvable, x = compute_x(C,index)
  return solvable,x
```

Abschließend können wir nun anhand der transformierten Matrix feststellen, ob das Gleichungssystem lösbar ist und, wenn ja, eine spezielle Lösung bestimmen.

```
def compute_x(C,ind):
# C=[A|b], wobei A eine obere Dreiecksmatrix mit m<= n ist.
# Wir loesen das System Ax=b unter Beruecksichtigung der
# Permutation index fuer x.
       .
       .
  if solvable:
    x[ind[d]]=C[d,n]/C[d,d]
    for i in range(1,d+1):              # for(i=1,i<d+1,i++)
      x[ind[d-i]]=C[d-i,n]
      for j in range(0,i):
        x[ind[d-i]]=x[ind[d-i]]-C[d-i,d-j]*x[ind[d-j]]
      x[ind[d-i]]=x[ind[d-i]]/C[d-i,d-i]
  return solvable,x
```

Bemerkung 2.1.2 *Grundsätzlich sollte man bei numerischen Berechnungen wegen der Rundungsfehler Abfragen nach Gleichheit vermeiden. So sollte für wirkliche Berechnungen in den oberen Routinen z.B. stets anstatt der Abfrage $x! = 0$ besser $|x| \geq \epsilon$ für eine kleine Konstante abgefragt werden.*

2.2 *LU*-Zerlegung

Wir wollen nun noch etwas detaillierter den Fall untersuchen, daß A ein reguläre Matrix ist. Speziell ist die Matrix dann quadratisch und wir definieren:

Definition 2.2.1 *Sei A eine $(n \times n)$ Matrix. A heißt* obere Dreiecksmatrix*, wenn $\forall 1 \leq i \leq n \forall i > j \leq n : A_{ij} = 0$ ist. Analog definieren, die* untere Dreiecksmatrix*, wenn $\forall 1 \leq i \leq n \forall i < j \leq n : A_{ij} = 0$ ist.*

Die folgenden Überlegungen sind nützlich, wenn das System $Ax = b$ mit festem A aber variierendem b zu lösen ist. Zunächst halten wir fest, daß wir uns bei regulärem A bei der Suche eines Pivotelementes auf die aktuelle Spalte beschränken können.

Proposition 2.2.2 *Sei A eine reguläre $(n \times n)$-Matrix und $1 \leq d \leq n$ so, daß für alle $0 < j < d$ und alle $i > j$ $A_{ij} = 0$ ist, d.h. A hat in den ersten $d - 1$ Spalten die Gestalt einer oberen Dreiecksmatrix. Dann gilt: $\exists d \leq i \leq n : A_{id} \neq 0$.*

Beweis. Wenn $\forall d \leq i \leq n : A_{id} = 0$, so haben die ersten d Spalten Nichtnulleinträge nur in den ersten $d-1$ Zeilen, sind also linear abhängig im Widerspruch zur angenommenen Regularität von A. $\square$

Unter der Annahme, daß nur Zeilenvertauschungen durchgeführt werden, können wir nun alle Transformationen, die im Gaußalgorithmus durchgeführt werden als Multiplikation des Systems $[A|b]$ von links mit einer geeigneten Matrix schreiben.

Definition 2.2.3 *Eine $(n \times n)$ Matrix der Gestalt*

$$P_{ij}^n = \begin{array}{r@{}l} & \begin{matrix} & & i & & & & j & & & \end{matrix} \\ \begin{matrix} \\ \\ i \\ \\ \\ \\ j \\ \\ \\ \\ \end{matrix} & \begin{pmatrix} 1 & & & & & & & & & \\ & 1 & & & & & & & 0 & \\ & & 0 & \cdots & \cdots & \cdots & 1 & & & \\ & & \vdots & \ddots & & & \vdots & & & \\ & & \vdots & & 1 & & \vdots & & & \\ & & \vdots & & & \ddots & \vdots & & & \\ & & 1 & \cdots & \cdots & \cdots & 0 & & & \\ & & & & & & & 1 & & \\ & 0 & & & & & & & \ddots & \\ & & & & & & & & & 1 \end{pmatrix} \end{array}$$

heißt Transpositionsmatrix. *Ein Produkt von Transpositionsmatrizen nennen wir eine* Permutationsmatrix.

Bemerkung 2.2.4 *Die Linksmultiplikation einer $(m \times n)$-Matrix A mit P_{ij}^m bewirkt die Vertauschung der i-ten mit der j-ten Zeile. Die Rechtsmultiplikation von A mit P_{ij}^n bewirkt die Vertauschung der i-ten mit der j-ten Spalte.*

Proposition 2.2.5 *a) Transpositionsmatrizen sind selbstinvers, d.h. $P_{ij}^n P_{ij}^n = I_n$.*

b) Eine $(n \times n)$-Matrix A ist eine Permutationsmatrix genau dann, wenn $A \in \{0,1\}^{n\times n}$ und für jedes $i \in \{1,\ldots,n\}$ ein $j_1 \in \{1,\ldots,n\}$ und ein $j_2 \in \{1,\ldots,n\}$ existieren mit $Ae_i = e_{j_1}$ und $e_i^\top A = e_{j_2}^\top$.

Beweis. Übung. $\square$

Auch ein Gaußeliminationsschritt läßt sich als Linksmultiplikation mit einer geeigneten Matrix schreiben.

Definition 2.2.6 *Eine $(n \times n)$ Matrix der Gestalt*

$$G_d = \begin{array}{cc} & \text{column } d \\ d & \begin{pmatrix} 1 & & & & \\ & \ddots & & 0 & \\ & & 1 & & \\ & & -g_{d+1,d} & & \\ & 0 & \vdots & \ddots & \\ & & -g_{n,d} & & 1 \end{pmatrix} \end{array}$$

heißt Frobeniusmatrix. *Sie unterscheidet sich nur in der d-ten Spalte von einer Einheitsmatrix. Wir können also schreiben*

$$G_d = I_n - g_d e_d^\top \ \textit{mit}\ g_d = (0, \ldots, 0, g_{d+1,d}, \ldots, g_{n,d}).$$

Ein Gaußeliminationsschritt bzgl. des Elementes a_{dd} wird bewirkt durch Linksmultiplikation mit einer Frobeniusmatrix, bei der $g_{id} = \frac{A_{id}}{A_{dd}}$.

Bemerkung 2.2.7 *Die Frobenius Matrix G_d ist regulär und man rechnet nach:*

$$G_d^{-1} = I_n + g_d e_d^\top = \begin{array}{cc} & \text{column } d \\ d & \begin{pmatrix} 1 & & & & \\ & \ddots & & 0 & \\ & & 1 & & \\ & & g_{d+1,d} & & \\ & 0 & \vdots & \ddots & \\ & & g_{n,d} & & 1 \end{pmatrix} \end{array}.$$

Proposition 2.2.8 *a) Ist P_{ij}^n eine Transpositionsmatrix mit $i, j > d$, so gilt*

$$P_{ij}^n G_d^{-1} P_{ij}^n = P_{ij}^n (I_n + g_d e_d^\top) P_{ij}^n = I_n + (P_{ij}^n g_d) e_d^\top.$$

b) Sind $P_{i_1 j_1}^n, \ldots, P_{i_k j_k}^n$ Transpositionsmatrizen mit $i_l, j_l > d$, so gilt

$$I_n + ((\prod_{l=1}^k P_{i_l j_l}^n) g_d) e_d^\top = \prod_{l=1}^k P_{i_l j_l}^n G_d^{-1} \prod_{l=1}^k P_{i_{k+1-l} j_{k+1-l}}^n.$$

Beweis. Übung. □

Führen wir den Gaußalgorithmus bei einer regulären Matrix A durch, so können wir dies auch durch eine Multiplikation mit einer Folge von Frobenius- und Transpositionsmatrizen beschreiben. Nach einer Gaußelimination können wir den Speicherplatz unterhalb des Diagonalelementes g_{dd} zur Abspeicherung des Vektors g_d nutzen. Der folgende Satz verdeutlicht, warum es sinnvoll ist, alle weiteren Zeilenvertauschungen auch mit dem g_d-Vektor durchzuführen. Genauer gilt:

Satz 2.2.9 (Satz von der Dreieckszerlegung) *Sei $A \in \mathbb{R}^{n\times n}$ eine reguläre Matrix und seien $P^n_{i_1 1}, \ldots P^n_{i_{n-1} n-1}$ und $G_1, \ldots, G_{n-1}$ die benötigten Transpositions- bzw. Frobeniusmatrizen. Setzen wir*

$$P = \prod_{k=1}^{n-1} P^n_{i_k k} \quad \textit{und} \quad L = \prod_{k=1}^{n-1} \left(I_n + \left(\prod_{j=1}^{n-k-1} P^n_{i_{n-j} n-j} \right) g_k e_k^\top \right)$$

und bezeichnen mit U die Transformierte von A, dann ist L eine untere Dreiecksmatrix und es gilt

$$PA = LU.$$

Beweis. Zunächst einmal ist $I_n + \left(\prod_{j=1}^{n-k-1} P^n_{i_{n-j} n-j} \right) g_k e_k^\top$ offensichtlich wieder eine Frobeniusmatrix. Somit ist L das Produkt von Frobeniusmatrizen mit von rechts nach links wachsendem Index, also (!) eine untere Dreiecksmatrix. Die Behauptung folgt nun, da $U = \prod_{k=1}^{n-1} G_{n-k} P^n_{i_{n-k} k} A$ und

$$\begin{aligned}
LU &= \prod_{k=1}^{n-1} \left(I_n + \left(\prod_{j=1}^{n-k-1} P^n_{i_{n-j} n-j} \right) g_k e_k^\top \right) \left(\prod_{k=1}^{n-1} G_{n-k} P^n_{i_{n-k} k} \right) A \\
&= \prod_{k=1}^{n-2} \left(I_n + \left(\prod_{j=1}^{n-k-1} P^n_{i_{n-j} n-j} \right) g_k e_k^\top \right) G^{-1}_{n-1} G_{n-1} \\
&\qquad P^n_{i_{n-1} n-1} \left(\prod_{k=2}^{n-1} G_{n-k} P^n_{i_{n-k} k} \right) A \\
&= \prod_{k=1}^{n-3} (\sim)(I_n + P^n_{i_{n-1} n-1} g_{n-2} e^\top_{n-2}) \\
&\qquad P^n_{i_{n-1} n-1} \left(\prod_{k=2}^{n-1} G_{n-k} P^n_{i_{n-k} k} \right) A \\
&= \prod_{k=1}^{n-3} (\sim) P^n_{i_{n-1} n-1} G^{-1}_{n-2} P^n_{i_{n-1} n-1} P^n_{i_{n-1} n-1} G_{n-2}
\end{aligned}$$

$$
\begin{aligned}
& P^n_{i_{n-2}n-2}\left(\prod_{k=3}^{n-1} G_{n-k}P^n_{i_{n-k}k}\right) A \\
= & \qquad \vdots \\
= & \prod_{k=1}^{n-d-1} (\sim) \left(\prod_{j=1}^{d-1} P^n_{i_{n-j}n-j}\right) G^{-1}_{n-d} \left(\prod_{j=1}^{d-1} P^n_{i_{n-d+j}n-d+j}\right) \\
& \left(\prod_{k=1}^{d-1} P^n_{i_{n-k}n-k}\right) G_{n-d}P^n_{i_{n-d}n-d} \left(\prod_{k=d+1}^{n-1} G_{n-k}P^n_{i_{n-k}k}\right) A \\
= & PA.
\end{aligned}
$$

□

Bemerkung 2.2.10 *Wir hatten bereits angedeutet, daß die LU-Zerlegung nützlich ist, wenn man $Ax = b$ für verschieden Vektoren b lösen muß. Dafür berechnet man zunächst $Lc = Pb$. Da L eine untere Dreiecksmatrix ist, läßt sich das durch eine einfache Rekursion lösen. Nun berechnet man $Ux = c$. Insgesamt gilt dann $Ax = P^{-1}PAx = P^{-1}LUx = P^{-1}Lc = P^{-1}Pb = b$.*

Analysieren wir die Komplexität des Algorithmus, so sind für den d-ten Gaußeliminationsschritt $(n-d+(n-d)^2)$ Multiplikationen und $(n-d)^2$ Additionen durchzuführen. Abgesehen von der Pivotsuche erält man einen Aufwand von $\frac{n^3}{3}-\frac{n}{3}$ Multiplikationen und $\frac{n^3}{3}-\frac{n^2}{2}+\frac{n}{6}$ Additionen. Bei der Lösung von $Ux = Pb$ beträgt der Aufwand $\binom{n}{2} = \frac{n^2}{2}-\frac{n}{2}$ Additionen und $\binom{n+1}{2} = \frac{n^2}{2}+\frac{n}{2}$ Multiplikationen.

Bemerkung 2.2.11 *Bei der Pivotsuche sollte man aus Gründen der numerischen Stabilität nach möglichst großen Einträgen suchen. Führt man allerdings eine Spaltenpivotsuche durch, so empfiehlt es sich, vorher das Maximum der Zeilen zu skalieren.*

2.3 Gauß-Jordan Algorithmus

Anstatt nur unterhalb des Pivotelementes die Variable zu eliminieren, kann man dies natürlich auch oberhalb der Diagonale tun. Wir erhalten so die Gauß-Jordan-Elimination.

```
def gaussjordanelim(A,i,j):
   A_ij=A[i,j]
   A[i,:]=A[i,:]/A_ij
```

```
for k in range(i)+range(i+1,m):
   A_kj=A[k,j]
   A[k,:]=A[k,:]-A[i,:]*A_kj
```

Analog zu dem Vorherigen kann man diese Eliminationsschritte zu einem Algorithmus kombinieren, bei dem in A eine Einheitsmatrix übrigbleibt und rechter Hand eine Lösung des Gleichungssystems steht.

Vorteile hat man beim Einsatz von Vektorrechnern, ansonsten ist die Komplexität um einen Faktor drei schlechter als eben. Den Gauß-Jordan-Eliminationsschritt werden wir allerdings in der Linearen Programmierung noch intensiv benutzen.

Auch den Gauß-Jordan-Eliminationsschritt kann man als Linksmultiplikation mit einer Matrix beschreiben. Dies hat bei eine Pivot auf dem Element a_{ij} die Gestalt

$$G_{i,j} = \begin{array}{cc} & \begin{array}{ccccc} & & i & & \end{array} \\ i & \begin{pmatrix} 1 & & \frac{-a_{1j}}{a_{ij}} & & \\ & \ddots & & 0 & \\ & & \frac{1}{a_{ij}} & & \\ & & \frac{-a_{i+1,j}}{a_{ij}} & & \\ & 0 & \vdots & \ddots & \\ & & \frac{-a_{nj}}{a_{ij}} & & 1 \end{pmatrix} \end{array}$$

und wird als η-Matrix bezeichnet. Die i-te Spalte wird auch η-Vektor genannt.

2.4 Wiederholung über Eigenwerte

Definition 2.4.1 *Sei A eine $(n \times n)$-Matrix über einem Körper $\mathbb{K}$. Eine Zahl $\lambda \in \mathbb{K}$ heißt* Eigenwert *von A, wenn es ein $x \in \mathbb{K}^n \setminus \{0\}$ gibt mit $Ax = \lambda x$. Jeder solche Vektor $x \neq 0$ heißt* Eigenvektor von A zum Eigenwert λ. *Ist A eine Matrix mit komplexwertigen Einträgen, so bezeichnen wir mit A^* die zu A transponierte, komplexkonjugierte Matrix. Ist $A = A^*$, so nennen wir A* hermitesch. *Eine Matrix Q heißt* orthogonal, *wenn $Q^\top Q = QQ^\top = I$.*

Eine reellwertige Matrix ist hermitesch genau dann, wenn sie symmetrisch ist.

Lemma 2.4.2 *Hermitesche Matrizen haben nur reelle Eigenwerte.*

Beweis. $Ax = \lambda x \Rightarrow x^*A^* = x^*A = \bar{\lambda}x^* \Rightarrow \lambda x^*x = x^*A^*x = \bar{\lambda}x^*x \Rightarrow \lambda = \bar{\lambda}$. □

Satz 2.4.3 *Ist A eine reellwertige, symmetrische Matrix, dann gibt es eine orthogonale Matrix Q, so daß $QAQ^\top = D$ eine Diagonalmatrix ist. Die Spalten von Q bilden eine* Orthonormalbasis aus Eigenvektoren.

Beweis. vgl. Mathematik für Wirtschaftsinformatiker, Kapitel V. □

Wir nennen $A = Q^\top DQ$ die *Hauptachsentransformation* von A, da die Eigenwerte von A die Diagonaleinträge von D sind und die Eigenvektoren die Zeilen von Q. Ist D eine Diagonalmatrix und y der Vektor der Diagonaleinträge, so schreiben wir auch $D = Diag(y)$.

2.5 Cholesky-Faktorisierung

Die Resultate des letzten Abschnitts besagen, daß man eine symmetrische Matrizen A durch Drehung und Spiegelung des Koordinatensystems in eine Diagonalmatrix überführen kann, welche die Eigenwerte von A auf der Diagonale hat. Sind diese alle nichtnegativ, so kann man aus A die „Wurzel ziehen". Solche Matrizen werden beim sogenannten Newton-Verfahren in der nichtlinearen Programmierung eine Rolle spielen.

Definition 2.5.1 *Sei $A \in \mathbb{R}^{n\times n}$ eine symmetrische Matrix. Dann heißt A* positiv definit *falls $x^\top Ax > 0$ für alle $x \in \mathbb{R}^n \setminus \{0\}$.*

Im folgenden werden wir zeigen, wie man bei positiv definiten Matrizen das Gleichungssystem $Ax = b$ noch effizienter lösen kann.

Proposition 2.5.2 *Eine reellwertige, symmetrische Matrix ist positiv definit genau dann, wenn der kleinste Eigenwert $\lambda_n > 0$ ist.*

Beweis. Die Bedingung offensichtlich notwendig, da aus $Ax = \lambda x$ sofort $x^\top Ax = \lambda\|x\|^2$ folgt. Umgekehrt, sei $b_1, \ldots, b_n$ eine Orthonormalbasis aus Eigenvektoren zu den Eigenwerten $\lambda_1 \geq \ldots \geq \lambda_n > 0$. Dann ist

$$\begin{aligned} x^\top Ax &= \left(\sum_{i=1}^n (x^\top b_i)b_i^\top\right) A \left(\sum_{j=1}^n (x^\top b_j)b_j\right) \\ &= \left(\sum_{i=1}^n (x^\top b_i)b_i^\top\right) \left(\sum_{j=1}^n (x^\top b_j)Ab_j\right) \end{aligned}$$

$$\begin{aligned}
&\overset{b_j \text{ sind EV}}{=} && \sum_{i,j=1}^{n} (x^\top b_i)(x^\top b_j)\lambda_j b_i^\top b_j \\
&\overset{b_i \text{ sind ONB}}{=} && \sum_{j=1}^{n} \lambda_i (x^\top b_j)^2 \\
&\overset{x\neq 0, \lambda_i > 0}{>} && 0.
\end{aligned}$$

□

Proposition 2.5.3 *Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit und $I \subseteq \{1, \dots n\}$. Dann ist auch $A_{I,I}$ symmetrisch und positiv definit.*

Beweis. Offensichtlich ist $A_{I,I}$ symmetrisch. Angenommen sie wäre nicht positiv definit, dann gäbe es ein $\tilde{x} \in \mathbb{R}^I \setminus \{0\}$ mit $\tilde{x}^\top A_{I,I}\tilde{x} \leq 0$. Wir definieren $x \in \mathbb{R}^n$ durch

$$x_i := \begin{cases} \tilde{x}_i & \text{falls } i \in I \\ 0 & \text{sonst.} \end{cases}$$

Dann ist $x \neq 0$ und $x^\top Ax = \tilde{x}^\top A_{I,I}\tilde{x} \leq 0$ im Widerspruch zur positiven Definitheit von A. □

Nach diesen Vorbereitungen können wir nun folgenden Satz zeigen.

Satz 2.5.4 (Satz von der Cholesky-Faktorisierung) *Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit. Dann existiert eine eindeutig bestimmte, reguläre untere Dreiecksmatrix L mit $A = LL^\top$ und $L_{ii} > 0$ für $i = 1, \dots, n$.*

Beweis. Wir führen vollständige Induktion über n. Im Fall $n = 1$ ist $A = (a_{11})$ mit $a_{11} > 0$ und wir setzen $L = (\sqrt{a_{11}})$. Sei also $n > 1$. Wir zerlegen A als

$$A = \begin{pmatrix} A_{n-1,n-1} & b \\ b^\top & a_{n,n} \end{pmatrix}.$$

Wie oben gezeigt ist $A_{n-1,n-1}$ positiv definit und symmetrisch. Nach Induktionsvoraussetzung gibt es also genau eine reguläre untere Dreiecksmatrix L_{n-1} mit positiven Diagonalelementen und $L_{n-1}L_{n-1}^\top = A_{n-1,n-1}$. Jedes L wie in dem Satz behauptet hat also notwendig die Form

$$L = \begin{pmatrix} L_{n-1,n-1} & 0 \\ c^\top & l_{n,n} \end{pmatrix}.$$

Setzen wir ein, so erhalten wir

$$A = \begin{pmatrix} A_{n-1,n-1} & b \\ b^\top & a_{n,n} \end{pmatrix} = \begin{pmatrix} L_{n-1} & 0 \\ c^\top & l_{n,n} \end{pmatrix} \begin{pmatrix} L_{n-1}^\top & c \\ 0 & l_{n,n} \end{pmatrix}$$

und somit als notwendige und hinreichende Bedingungen $L_{n-1}c = b$ und $\|c\|^2 + l_{n,n}^2 = a_{n,n}$. L_{n-1} ist regulär also können wir $c = L_{n-1}^{-1}b$ setzen. Die Behauptung folgt nun, wenn wir zeigen können, daß $a_{n,n} - \|c\|^2 > 0$ ist. Dafür betrachten wir $x^\top = (c^\top L_{n-1}^{-1}, -1)$. Dann ist $x \neq 0$ und somit

$$\begin{aligned}
0 &< x^\top A x \\
&= x^\top \begin{pmatrix} A_{n-1,n-1} & b \\ b^\top & a_{n,n} \end{pmatrix} \begin{pmatrix} (L_{n-1}^{-1})^\top c \\ -1 \end{pmatrix} \\
&= x^\top \begin{pmatrix} L_{n-1}L_{n-1}^\top (L_{n-1}^{-1})^\top c - b \\ c^\top L_{n-1}^\top (L_{n-1}^{-1})^\top c - a_{n,n} \end{pmatrix} \\
&= (c^\top L_{n-1}^{-1}, -1) \begin{pmatrix} 0 \\ c^\top c - a_{n,n} \end{pmatrix} \\
&= a_{n,n} - \|c\|^2.
\end{aligned}$$

□

Für die Berechnung der Cholesky-Faktorisierung betrachten wir das Zustandekommen der einzelnen Einträge von A.

$$a_{ij} = L_{i.} L_{.j}^\top = \sum_{k=1}^{n} l_{ik} l_{jk} = \sum_{k=1}^{\max\{i,j\}} l_{ik} l_{jk}.$$

Wir können daraus leicht rekursiv die Einträge von L ausrechnen und erhalten folgenden Algorithmus:

```
def cholesky(A):
  n=A.shape[0]
  L=zeros((n,n))
  try:
    for k in range(n):
      L[k,k]=sqrt(A[k,k]-dot(L[k,:],L[k,:]))
      for l in range(k+1,n):
        L[l,k]=(A[l,k]-dot(L[l,:],L[k,:]))/L[k,k]
  except:
    print "Matrix nicht positiv definit"
  return L
```

Analysieren wir den Rechenaufwand des Verfahrens, so haben wir bei festem Zeilenindex l einen Aufwand von $\sum_{i=1}^{l-1} i = \binom{l}{2}$ Additionen, $\binom{l}{2}$ Multiplikationen, $l-1$ Divisionen und einer Quadratwurzel. Aufsummiert erhalten wir $\frac{1}{2}\sum_{l=1}^{n} l^2 - l = \frac{n(n+1)(2n+1)-3n(n+1)}{12} = \frac{n^3-n}{6}$ Additionen und Multiplikationen, $\binom{n}{2}$ Divisionen und n Quadratwurzeln. Die Cholesky-Faktorisierung läßt sich analog zur LU-Zerlegung zur Lösung von Gleichungssystemen $Ax = b$ benutzen. Der Aufwand beträgt aber nur etwa die Hälfte des Gaußverfahrens. Außerdem ist dieser Algorithmus numerisch stabiler.

Beispiel 2.5.5 *Wir betrachten die Matrix*

$$A = \begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & 2 & 0 & 0 \\ -1 & 0 & 3 & 1 \\ -1 & 0 & 1 & 4 \end{pmatrix}$$

und berechnen

$$\begin{aligned}
&l_{1,1} = \sqrt{1},\; l_{2,1} = -1,\; l_{3,1} = -1,\; l_{4,1} = -1,\\
&l_{2,2} = \sqrt{2-(-1)^2} = 1,\; l_{3,2} = (0-(-1)*(-1))/1 = -1,\\
&l_{4,2} = (0-(-1)*(-1))/1 = -1,\\
&l_{3,3} = \sqrt{3-1-1} = 1,\; l_{4,3} = (-1-(-1)*(-1)-(-1)*(-1))/1,\\
&l_{4,4} = \sqrt{4-1-1-1} = 1
\end{aligned}$$

also

$$\begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & 2 & 0 & 0 \\ -1 & 0 & 3 & 1 \\ -1 & 0 & 1 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & -1 & 1 & 0 \\ -1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & -1 & -1 & -1 \\ 0 & 1 & -1 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

.

2.6 Matrixnormen

Im diesem Abschnitt wollen wir Überlegungen zur numerischen Stabilität von Operationen der Linearen Algebra beginnen. In der Mathematik für Chemiker und Wirtschaftsinformatiker ist bereits Fehlerfortpflanzung von rellwertigen Funktionen besprochen worden. Wir erinnern daran, daß dort die Linearisierung (Ableitung) der Funktion eine Maßzahl für die numerische Problematik des Ausdrucks gab. Wie sieht das im Mehrdimensionalen aus? Dafür werden wir Normen von Matrizen studieren.

Definition 2.6.1 *Sei X ein Vektorraum über $\mathbb{R}$. Eine Abbildung $\|\cdot\| : X \to \mathbb{R}$ heißt* Norm, *wenn*

(N1) $\|x\| = 0 \Leftrightarrow x = 0$,

(N2) $\forall \alpha \in \mathbb{R} : \|\alpha x\| = |\alpha| \|x\|$; (Homogenität)

(N3) $\|x + y\| \leq \|x\| + \|y\|$; Dreiecksungleichung).

Ein Vektorraum mit Norm heißt normierter Vektorraum.

Wegen $0 = \|x - x\| \leq \|x\| + \|-x\| = 2\|x\|$ gilt stets $\|x\| \geq 0$. Zu einem normierten n-dimensionalen $\mathbb{R}$-Vektorraum X und einem normierten m-dimensionalen $\mathbb{R}$-Vektorraum Y gibt es eine natürliche Norm auf dem Vektorraum der $(m \times n)$-Matrizen über $\mathbb{R}$.

Proposition 2.6.2 *Seien X, Y normierte $\mathbb{R}$-Vektorräume der Dimension n bzw. m mit Normen $\|\cdot\|_X, \|\cdot\|_Y$. Dann ist die Abbildung $\|\cdot\|_{X\to Y} : \mathbb{R}^{m\times n} \to \mathbb{R}$ definiert durch*

$$\|A\|_{X\to Y} := \sup_{x\in\mathbb{R}^n\setminus\{0\}} \frac{\|Ax\|_Y}{\|x\|_X} \stackrel{!}{=} \max_{\|x\|_X=1} \|Ax\|_Y$$

eine Norm auf dem Vektorraum der linearen Abbildungen von X nach Y, die natürliche Norm.

Beweis. ad N1: $\max_{\|x\|_X=1} \|Ax\|_Y = 0$ impliziert mit der Homogenität, daß $Ax = 0$ für alle $x \in \mathbb{R}^n$. Also stellt A die Nullabbildung dar und somit haben wir $A = 0$.

ad N2: $\max_{\|x\|_X=1} \|\alpha Ax\|_Y = \max_{\|x\|_X=1} |\alpha|\|Ax\|_Y = |\alpha| \max_{\|x\|_X=1} \|Ax\|_Y$.

ad N3:

$$\begin{aligned} \max_{\|x\|_X=1} \|(A+B)x\|_Y &\leq \max_{\|x\|_X=1} (\|Ax\|_Y + \|Bx\|_Y) \\ &\leq \max_{\|x\|_X=1} \|Ax\| + \max_{\|y\|_X=1} \|By\|_Y. \end{aligned}$$

□

Wegen $\|A\|_{X\to Y} \geq \|Ax\|_Y/\|x\|_X$ gilt nun stets $\|Ax\|_Y \leq \|A\|_{X\to Y}\|x\|_X$. Für natürliche Matrixnormen von Endomorphismen (linearen Abbildungen innerhalb eines festen Vektorraumes X) gilt $\|I\| = 1$ und $\|AB\| \leq \|A\|\|B\|$. Letzteres folgt aus

$$\|ABx\|_X \leq \|A\|_{X\to X}\|Bx\|_X \leq \|A\|_{X\to X}\|B\|_{X\to X}\|x\|_X.$$

Beispiel 2.6.3 *Wir betrachten die im letzten Kapitel bereits eingeführten Normen $\|\cdot\|_1, \|\cdot\|_2, \|\cdot\|_\infty$. Sind dann $X = \mathbb{R}^n = Y$, so schreiben wir für die zugehörigen Matrixnormen kurz ebenso $\|A\|_1, \|A\|_2, \|A\|_\infty$. Es gilt:*

a) $\|A\|_1 = \max_{j\in\{1,\ldots n\}} \sum_{i=1}^n |a_{ij}|$ *(Spaltensummennorm),*

b) $\|A\|_\infty = \max_{i\in\{1,\ldots n\}} \sum_{j=1}^n |a_{ij}|$ *(Zeilensummennorm),*

c) $\|A\|_2 = \max\{\sqrt{\lambda} \mid \lambda$ *ist Eigenwert von* $A^\top A\}$ *(Spektralnorm).*

Beweis. *Zum Beweis von a). sei x mit* $\|x\|_1 = \sum_{j=1}^n |x_j| = 1$. *Dann gilt*

$$\begin{aligned}
\|Ax\|_1 &= \sum_{i=1}^n \left| \sum_{j=1}^n a_{ij}x_j \right| \\
&\leq \sum_{i,j=1}^n |a_{ij}||x_j| \\
&= \sum_{j=1}^n \left(|x_j| \sum_{i=1}^n |a_{ij}| \right) \\
&\leq \sum_{j=1}^n \left(|x_j| \max_{k\in\{1,\ldots,n\}} \sum_{i=1}^n |a_{ik}| \right) \\
&\overset{\|x\|_1=1}{=} \max_{j\in\{1,\ldots,n\}} \sum_{i=1}^n |a_{ij}|.
\end{aligned}$$

Die erste Behauptung folgt nun aus $\|Ae_j\|_1 = \sum_{i=1}^n |a_{ij}|$.

ad b).: Sei nun x mit $\|x\|_\infty = \max_{j\in\{1,\ldots,n\}} |x_j| = 1$. *Dann gilt*

$$\begin{aligned}
\|Ax\|_\infty &= \max_{i\in\{1,\ldots,n\}} \left| \sum_{j=1}^n a_{ij}x_j \right| \\
&\leq \max_{i\in\{1,\ldots,n\}} \left\{ \sum_{j=1}^n |a_{ij}||x_j| \right\} \\
&\overset{|x_j|\leq 1}{\leq} \max_{i\in\{1,\ldots,n\}} \sum_{j=1}^n |a_{ij}|.
\end{aligned}$$

Die Behauptung folgt somit, wenn man den $(1,-1)$*-Vektor betrachtet, der die gleichen Vorzeichen hat wie die Zeile, in der das Maximum angenommen wird.*

Für den Beweis von c) erinnern wir uns zunächst daran, daß es, da $A^\top A$ symmetrisch ist, es eine Orthonormalbasis $b_1, \ldots, b_n$ aus Eigenvektoren zu Eigenwerten $\lambda_1 \geq \ldots \geq \lambda_n \overset{!}{\geq} 0$ von $A^\top A$ gibt. Sei nun $x = \sum_{i=1}^n \beta_i b_i$ mit $\|x\|_2 = \sqrt{x^\top x} = \sqrt{\sum_{i=1}^n \beta_i^2} = 1$. Dann ist

$$\begin{aligned}
\|Ax\|_2^2 &= x^\top A^\top Ax \\
&= (\sum_{i=1}^n \beta_i b_i^\top) A^\top Ax \\
&= (\sum_{i=1}^n \beta_i \lambda_i b_i^\top)(\sum_{j=1}^n \beta_j b_j) \\
&= \sum_{i=1}^n \beta_i^2 \lambda_i \\
&\leq \lambda_1.
\end{aligned}$$

Die Behauptung folgt nun aus $b_1^\top A^\top A b_1 = \lambda_1 b_1^\top b_1$. □

2.7 Kondition

Die motivierende Fragestellung dieses Abschnitts ist: Wie wirken sich Datenfehler bei der Aufgabe $Ax = b$ mit einer regulären Matrix A (bei exakter Lösung) auf den Vektor x aus. In diesem Kapitel gehen wir davon aus, daß eine feste Vektornorm $\|\cdot\|_{\mathbb{R}^n}$ mit zugehöriger Matrixnorm $\|\cdot\|_{\mathbb{R}^n \to \mathbb{R}^n}$ gegeben ist.

Beschränken wir uns zunächst auf eine fehlerbehaftete rechte Seite

$$A(x + \Delta x) = b + \Delta b.$$

Hieraus ergibt sich $\Delta x = A^{-1} \Delta b$ und somit

$$\|\Delta x\| \leq \|A^{-1}\| \, \|\Delta b\|.$$

Mit $\|b\| \leq \|A\| \, \|x\|$ erhalten wir $\|x\| \geq \frac{\|b\|}{\|A\|}$ und hieraus folgende Abschätzung für den relativen Fehler

$$\frac{\|\Delta x\|}{\|x\|} \leq \|A^{-1}\| \, \|A\| \frac{\|\Delta b\|}{\|b\|}.$$

Definition 2.7.1 *Sei $A \in \mathbb{R}^{n \times n}$ eine reguläre Matrix. Die Zahl $cond(A) := \|A^{-1}\| \, \|A\|$ heißt* Kondition *der Matrix A.*

Die Kondition ist abhängig von der gewählten Norm. Für die natürliche Matrixnorm eines normierten Raumes gilt

$$\operatorname{cond}(A) = \|A^{-1}\|\,\|A\| \geq \|A^{-1}A\| = \|I_n\| = 1.$$

Um auch Auswirkungen von Störungen von A abschätzen zu können, beweisen wir zunächst:

Lemma 2.7.2 *Sei $A \in \mathbb{R}^{n\times n}$ und $\|A\| < 1$. Dann ist $I_n + A$ regulär und*

$$\frac{1}{1+\|A\|} \leq \|(I_n + A)^{-1}\| \leq \frac{1}{1-\|A\|}.$$

Beweis. Wegen $\|x + Ax - Ax\| \leq \|x + Ax\| + \|Ax\|$ gilt:

$$\|(I_n + A)x\| = \|x + Ax\| \geq \|x\| - \|A\|\|x\| \geq (1 - \|A\|)\|x\|.$$

Folglich kann $(I_n + A)x = 0$ nur gelten, wenn $x = 0$ ist, somit muß $I_n + A$ regulär sein. Ferner haben wir

$$1 \leq \|(I_n + A)^{-1}\|\,\|I_n + A\| \overset{(N3)}{\leq} \|(I_n + A)^{-1}\|(1 + \|A\|)$$

und

$$\begin{aligned}
\|(I_n + A)^{-1}\| &= \|(I_n + A)^{-1} + (I_n + A)^{-1}A - (I_n + A)^{-1}A\| \\
&\leq \|(I_n + A)^{-1}(I_n + A)\| + \|(I_n + A)^{-1}A\| \\
&\leq 1 + \|(I_n + A)^{-1}\|\,\|A\|
\end{aligned}$$

also

$$\|(I_n + A)^{-1}\|(1 - \|A\|) \quad \leq \quad 1.$$

□

Wir sind an den Auswirkungen einer Störung von A bei der Lösung von $Ax = b$ und damit an einer Abschätzung von $\operatorname{cond}(A + \Delta A)$ interessiert. Wir folgern deshalb:

Lemma 2.7.3 (Störungslemma) *Seien $A, B \in \mathbb{R}^{n\times n}$, A regulär und $\|A^{-1}\|\,\|B - A\| < 1$. Dann ist auch B regulär und*

$$\|B^{-1}\| \leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\|\,\|B - A\|}.$$

Beweis. Nach Voraussetzung ist $\|A^{-1}(B-A)\| \leq \|A^{-1}\| \, \|(B-A)\| < 1$. Nach Lemma 2.7.2 ist also $(I_n + A^{-1}B - I_n)$ regulär, also auch B regulär, und es gilt

$$\begin{aligned} \|B^{-1}\| \leq \|B^{-1}A\| \, \|A^{-1}\| &\leq \|A^{-1}\| \frac{1}{1 - \|A^{-1}B - I_n\|} \\ &\leq \|A^{-1}\| \frac{1}{1 - \|A^{-1}\| \, \|B-A\|}. \end{aligned}$$

□

Nun können wir abschließend folgenden Satz beweisen.

Satz 2.7.4 *Seien $A, \Delta A \in \mathbb{R}^{n\times n}$ und gelte $\|A^{-1}\| \, \|\Delta A\| < 1$. Seien x bzw. $x + \Delta x$ Lösungen des Systems $Ax = b$ bzw. $(A + \Delta A)(x + \Delta x) = b$. Dann läßt sich der relative Fehler abschätzen durch*

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{cond\,(A)}{1 - cond(A)\frac{\|\Delta A\|}{\|A\|}} \frac{\|\Delta A\|}{\|A\|}.$$

Beweis. Nach Voraussetzung ist $\|A^{-1}\| \, \|A + \Delta A - A\| < 1$. Also ist nach dem Störungslemma $A + \Delta A$ regulär und

$$\|(A + \Delta A)^{-1}\| \leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\| \, \|\Delta A\|}.$$

Aus $(A + \Delta A)(x + \Delta x) - Ax = 0$ schließen wir $\Delta x = -(A + \Delta A)^{-1}\Delta A x$. Durch Einsetzen erhalten wir

$$\begin{aligned} \|\Delta x\| &\leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\| \, \|\Delta A\|} \|\Delta A\| \, \|x\| \\ &= \frac{\|A^{-1}\| \, \|A\|}{1 - \|A^{-1}\| \|A\| \frac{\|\Delta A\|}{\|A\|}} \frac{\|\Delta A\|}{\|A\|} \|x\|, \end{aligned}$$

woraus die Behauptung folgt □