Vorlesungsskript
Numerische Mathematik I

F. Natterer

Institut für Numerische
und instrumentelle Mathematik

WS 1998/99, Di/Fr 11-13, M 4

Inhaltsverzeichnis

Kapitel 1

Lineare Gleichungssysteme

1.1 Das Eliminationsverfahren

Sei K der Körper der komplexen oder der Körper der reellen Zahlen. A sei eine (n, n)-Matrix über K und b ein Vektor der Länge n. Wir betrachten das lineare Gleichungssystem

$$Ax = b \tag{1.1}$$

für den n-Vektor x. Aus den Anfangsvorlesungen ist bekannt, daß (1.1) genau dann für alle b eindeutig lösbar ist, wenn det $(A) \neq 0$. Es gibt auch eine explizite Formel für die Lösung x, nämlich die Cramer'sche Regel: Sei A_j die Matrix, die aus A durch Ersetzen der j-ten Spalte mit b entsteht. Dann gilt für die j-te Komponente x_j von x

$$x_j = \frac{\det(A_j)}{\det(A)} \quad , \qquad j = 1, \ldots, n \; . \tag{1.2}$$

Dies ist eine für den Mathematiker vollkommen befriedigende Lösung des Problems. Sie ist vollkommen einfach und explizit, und gilt immer dann, wenn das Problem sinnvoll ist, d.h. wenn det $(A) \neq 0$.

Für die Numerik ist (1.2) aber vollkommen nutzlos. Der Rechenaufwand zur Auswertung von (1.2) ist nämlich viel zu hoch. Die Berechnung von det (A) nach der Definition der Determinante als Summe von $n!$ Produkten mit je n Faktoren (wir werden später schnellere Methoden kennenlernen) erfordert $n!(n-1)$ Multiplikationen und $n! - 1$ Additionen/Subtraktionen.

Das gleiche gilt für det (A_j). Also benötigen wir zur Auswertung von (1.2)

$$(n+1)(n!(n-1) \text{ Mult. } + (n!-1) \text{ Add./Sub.}) + n \text{ Divisionen} \tag{1.3}$$

In Komplexitätsbetrachtungen dieser Art interessieren uns nur die Terme höchster Ordnung in n. Die anderen fassen wir durch das O-Symbol (O für Ordnung) zusammen. Wir schreiben $\mathrm{O}(g)$ für jeden Ausdruck, der sich in der Form

$$|\mathrm{O}(g)| \leq C|g| , \qquad g \to \infty$$

abschätzen läßt mit einer von g unabhängigen Konstanten C. Für (1.3) können wir dann schreiben

$$(n+2)! \text{ Mult. } + \mathrm{O}((n+1)!) \text{ weitere Operationen .} \tag{1.4}$$

Man interessiert sich dann nur für den ersten Term $(n+2)!$, nimmt also an, n sei so groß, daß dieser Term überwiegt. Für $n = 20$ ist $(n+2)! = 1.1\ \ 10^{21}$, eine für die praktische Rechnung ganz absurde Zahl.

An dieser Stelle wollen wir gleich unsere Notation für Dezimalzahlen oder Gleitkommazahlen (*floating point*) erläutern. Mit 1.1 z.B. meinen wir eine reelle Zahl, welche nach Rundung auf 1 Stelle hinter dem Dezimalpunkt 1.1 ergibt. Die Schreibweise $x = 1.1$ meint also, daß $1.05 \leq x < 1.15$. Wir wollen diese Vereinbarung aber nicht allzu pedantisch verstehen, den Begriff der Rundung also nicht ganz präzisieren.

Wir werden nun das Eliminationsverfahren zur Lösung von (1.1) besprechen. Dieses wurde zwar in der Schule und in den Anfängervorlesungen schon behandelt. Es gibt aber spezielle numerische Aspekte, die uns zwingen, noch einmal ganz von vorne anzufangen. Wir schreiben dazu (1.1) ausführlich in der Form

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 \quad &+ \cdots + \quad a_{1,n}x_n = b_1 \ , \\ a_{2,1}x_1 + a_{2,2}x_2 \quad &+ \cdots + \quad a_{2,n}x_n = b_2 \ , \\ &\cdots \\ a_{n,1}x_1 + a_{n,2}x_2 \quad &+ \cdots + \quad a_{n,n}x_n = b_n \ . \end{aligned} \tag{1.5}$$

Sei det $(A) \neq 0$. Dann ist mindestens ein $a_{i,1} \neq 0$.

Eventuell nach einer Zeilenvertauschung können wir $a_{1,1} \neq 0$ annehmen. Wir eliminieren nun aus den Gleichungen 2 bis n die Variable x_1, indem wir zu den Zeilen von 2 bis n das $\ell_{i,1} = a_{i,1}/a_{11}$-fache der ersten Zeilen subtrahieren, $i = 2, \ldots, n$. Es entsteht dann das zu (1.5) äquivalente System

$$
\begin{array}{lllll}
a_{1,1}x_1 + & a_{1,2}x_2 & + \cdots + & a_{1,n}x_n & = b_1 \\
& (a_{2,2} - \ell_{2,1}a_{1,2})x_2 & + \cdots + & (a_{2,n} - \ell_{2,1}a_{1,n})x_n & = b_2 - \ell_{2,1}b_1 \\
& & \cdots & & \\
& (a_{n,2} - \ell_{n,1}a_{1,2})x_2 & + \cdots + & (a_{2,n} - \ell_{n,1}a_{1,n})x_n & = b_n - \ell_{n,1}b_1
\end{array}
\tag{1.6}
$$

Es enthält in den Zeilen $2, \ldots, n$ nur noch die Variablen $x_2, \ldots, x_n$. Diese Zeilen bilden also ein lineares System von $n-1$ Gleichungen in $n-1$ Unbekannten. Wir schreiben es als

$$
\begin{array}{llll}
a_{2,2}^{(2)}x_2 & + \cdots + & a_{2,n}^{(2)}x_n & = b_2^{(2)} \\
& \cdots & & \\
a_{n,2}^{(2)}x_2 & + \cdots + & a_{n,n}^{(2)}x_n & = b_n^{(2)}
\end{array}
\tag{1.7}
$$

$$
\begin{array}{rcllll}
a_{i,k}^{(2)} & = & a_{i,k} - \ell_{i,1}a_{1,k} & , & i, k = 2, \ldots, n & , \\
b_i^{(2)} & = & b_i - \ell_{i,1}b_1 & , & i = 2, \ldots, n & , \\
\ell_{i,1} & = & a_{i,1}/a_{1,1} & , & i = 2, \ldots, n & .
\end{array}
$$

Ist (1.7) gelöst, so bekommt man eine Lösung von (1.6) (und damit von (1.5) oder (1.1)) durch

$$x_1 = (b_1 - a_{1,2}x_2 - \cdots - a_{1,n}x_n)/a_{11} \; .$$

Das lineare System (1.7) wird nun genauso behandelt wie (1.5). Man eliminiert x_2 aus den Gleichungen $3, \ldots, n$ - eventuell nach Zeilenvertauschung - und bekommt dann ein lineares System von $n-2$ Gleichungen für die $n-2$ Unbekannten $x_3, \ldots, x_n$. So geht man rekursiv vor, bis man bei einem System mit nur einer Unbekannten ankommt, das man direkt löst.

Wir wollen für das Eliminationsverfahren ein rekursives Programm schreiben:

```
elim (A, b, x, j, n)
/* Löst  Σ_{k=j}^{n} a_{i,k} x_k = b_i ,   i = j, ..., n  */
{   Zeilenvertauschung () ;
    for   i = j + 1, ..., n
      {   ℓ = a_{i,j} / a_{j,j} ;
          for  k = j + 1, ..., n   a_{i,k} = a_{i,k} − ℓ * a_{j,k} ;
          b_i = b_i − ℓ * b_j ;
      }
    if   (j < n) elim (A, b, x, j + 1, n);
    x_j = (b_j − a_{j,j+1} * x_{j+1} − ··· − a_{j,n} * x_n) / a_{j,j};
}
```

Die Lösung des Systems (1.1) geschieht nun einfach durch den Aufruf

$$\mathrm{elim}(A, b, x, 1, n)\ ;$$

Die in diesem Programm noch nicht spezifizierte Prozedur Zeilenvertauschung sucht unter den Elementen $a_{i,j}$, $i \geq j$ ein von Null verschiedenes, etwa $a_{i_0,j}$, und vertauscht dann die Zeilen j und i_0. Das Element $a_{i_0,j}$ heißt dann das Pivot, Zeile i_0 die Pivotzeile, und Spalte j die Pivotspalte. Also:

```
Zeilenvertauschung ( );

{   Suche  i_0;
    for k = j, ..., n     swap (a_{i_0,k}, a_{j,k});
    swap (b_{i_0}, b_j);
}
```

In der Regel wird man mehrere i_0 zur Auswahl haben. Rein mathematisch gesehen ist es gleichgültig, welches wir nehmen. Für die Numerik ist es aber entscheidend, daß man aus den möglichen Indizes i denjenigen auswählt, für den $|a_{i,j}|$ maximal ist. Man bestimmt also $i_0 \geq j$ so, daß $|a_{i_0,j}| \geq |a_{i,j}|$, $i = j, \ldots, n$. Dieses i_0 ist im allgemeinen immer noch nicht eindeutig bestimmt, aber das spielt nun auch numerisch keine Rolle mehr. Wir sprechen von maximaler Spaltenpivotsuche.

Um zu verstehen, warum es gut ist, das Pivotelement möglichst groß zu wählen, müssen wir uns Gedanken über die Entstehung und Fortspflanzung von Rundungsfehlern machen.

Sei x eine reelle Zahl und $\tilde{x}$ eine Näherung für x. Dann nennen wir $|x - \tilde{x}|$ den absoluten und $|(x - \tilde{x})/x|$ den relativen Fehler von $\tilde{x}$, letzteres natürlich nur, wenn $x \neq 0$. Ist $\tilde{x}$ ein gerundeter Wert von x, so sprechen wir von den absoluten bzw. relativen Rundungsfehlern.

Aus einem kleinen Fehler am Anfang kann ein großer Fehler am Ende werden (so schon Aristoteles im 5. Buch der Metaphysik). Betrachten wir einmal folgende 4-stellige Rechnung

$$1.2547 - 1.2534 = 0.0013\ .$$

Die beiden Operanden haben gemäß unserer Schreibweise je einen absoluten Fehler $\leq 5 \cdot 10^{-5}$. Daraus wird im Resultat ein absoluter Fehler von maximal 10^{-4}. Für die relativen Fehler bedeutet das ein gewaltiges Anwachsen. Aus $4 \cdot 10^{-5}$ in den Operanden wird $8 \cdot 10^{-2}$ im Resultat. Wir sehen, daß bei der Subtraktion nahezu gleichgroßer Zahlen der relative Fehler stark ansteigt. Wir sprechen von Auslöschung. Alle anderen Operationen, also Addition (von Zahlen gleichen Vorzeichens), Multiplikation und Division, führen zu keiner Verstärkung des relativen Fehlers. Auslöschung ist das Problem Nr. 1 der Numerik.

Glücklicherweise verursachen die meisten Auslöschungen keine Probleme. Bei der Analyse und dem Entwurf numerischer Algorithmen müssen Auslöschungen erkannt und - durch geeignete Gestaltung des Algorithmus - unschädlich gemacht werden. Dies wollen wir an Hand des Eliminationsverfahrens einmal vorführen.

Die einzige Art von Rechenoperationen, die beim Eliminationsverfahren ausgeführt wird, ist

$$a_{i,k} - \ell_{i,j} a_{j,k}\ , \quad \ell_{i,j} = a_{i,j}/a_{j,j}\ .$$

Wir wollen annehmen, daß alle Elemente von A die gleiche Größenordnung haben, etwa 1. Auslöschung bedeutet dann, daß ein sehr kleines $a_{i,k}$ auftritt. Dieses hat dann notwendigerweise großen relativen Fehler. Wir unterscheiden drei Fälle:

1. Das kleine Element $a_{i,k}$ steht nicht in Spalte j. Dann kann $a_{i,k} - \ell_{i,j} a_{j,k}$

trotz des großen relativen Fehlers von $a_{i,k}$ genau berechnet werden, da $a_{i,k}$ im Vergleich zu $\ell_{i,j}a_{j,k}$ sehr klein ist. Die Auslöschung ist harmlos.

2. Das Element $a_{i,j}$ in der j-ten Spalte unterhalb des Elementes $a_{j,j}$ fällt klein aus. Dann ist $\ell_{i,j}$ klein und hat großen relativen Fehler. $a_{i,k} - \ell_{i,j}a_{j,k}$ kann aber trotzdem genau berechnet werden, weil $\ell_{i,j}a_{j,k}$ klein gegenüber $a_{i,k}$ ist. Wieder ist die Auslöschung harmlos.

3. $a_{j,j}$ fällt klein aus. Jetzt wird $\ell_{i,j}$ groß und falsch. $a_{i,k} - \ell_{i,j}a_{j,k}$ kann nicht mehr genau berechnet werden. Die Auslöschung ist nicht harmlos.

Der Auslöschungseffekt ist um so größer, je kleiner das Pivot $a_{j,j}$ ist. Im Prinzip könnte man jedes Element unterhalb und rechts von $a_{j,j}$ als Pivot nehmen. Dies würde Zeilen- und Spaltenvertauschungen erfordern. Die Praxis und auch die mathematische Theorie zeigen aber, daß es genügt, das betragsgrößte Element unterhalb von $a_{j,j}$ zu wählen. So kommt man zur maximalen Spaltenpivotsuche.

Zeilenvertauschung ( );

```
{   piv = |a_jj|;  i_0 = j;
    for i = j+1, ..., n
    if (|a_i,j| > piv)    {piv = |a_i,j|;  i_0 = i ; }
    if (piv  < ε)    {print ("Matrix singulär"); exit (1);}
    for k = j, ..., n     swap (a_i0,k, a_j,k);
    swap (b_i0, b_j);
}
```

Unser Programm für das Eliminationsverfahren enthält einen rekursiven Prozeduraufruf. Das ist zwar elegant, aber ineffizient. Auflösen der Rekursion führt zum endgültigen Programm.

```
elim (A, b, x, n)        /⋆ Löst   Ax = b ⋆/
  for  j = 1, ..., n
    {   Zeilenvertauschung ( ) ;
        for   i = j + 1, ..., n
          {  ℓ = a_ij / a_jj  ;
             for  k = j + 1, ..., n   a_ik = a_ik − ℓ ∗ a_jk  ;
             b_i = b_i − ℓ ∗ b_j  ;
          }
    }
  for  j = n, ..., 1
    { x_j = b_j;
      for  k = j + 1, ..., n     x_j = x_j − a_j,k ∗ x_k;
      x_j = x_j / a_j,j;
    }
```

Wir wollen die Anzahl der Rechenoperationen beim Eliminationsverfahren bestimmen. Da eine Multiplikation meist zusammen mit einer Add./Sub. auftritt, vereinbart man

1 flop   (floating point operation) = 1 Mult./Div. + 1 Add./Sub.   .

Sei K_j die Anzahl der flops für den j-ten Eliminationsschritt. Es ist

$$K_j = (n-j)^2 + \mathrm{O}(n-j) \ ,$$

also

$$\sum_{j=1}^{n-1} K_j = \sum_{j=1}^{n-1} \Big((n-j)^2 + \mathrm{O}(n-j) \Big) = \frac{1}{3} n^3 + \mathrm{O}(n^2) \ .$$

Man vergleiche dies mit der Anzahl der Operationen (1.4) für die Cramer'sche Regel.

1.2 Die LR-Zerlegung

Wir wollen nun den Eliminationsprozess in einer anderen Form darstellen. Wir nennen die (n,n)-Matrix $L = (\ell_{i,k})$ linke Dreiecksmatrix, wenn $\ell_{i,k} = 0$ ist für $k > i$. Ebenso nennen wir die (n,n)-Matrix $R = (r_{i,k})$ rechte Dreiecksmatrix, wenn $r_{i,k} = 0$ ist für $i > k$. Die invertierbaren linken (und auch rechten) Dreiecksmatrizen bilden eine Gruppe bezüglich der Multiplikation.

Unter der LR-Zerlegung einer (n,n)-Matrix A versteht man die Berechnung von Dreiecksmatrizen L, R mit $A = LR$. Ist die LR-Zerlegung hergestellt, kann $Ax = b$ durch

$$Ly = b \quad , \quad Rx = y$$

ersetzt werden. Diese Systeme mit Dreiecksmatrizen können durch sogenanntes Vorwärts- bzw. Rückwärtseinsetzen gelöst werden:

$$\begin{array}{lcllcl}
y_1 &=& b_1/\ell_{1,1} \ , & x_n &=& y_n/r_{n,n} \ , \\
y_2 &=& (b_2 - \ell_{2,1}y_1)/\ell_{2,2} \ , & x_{n-1} &=& (y_{n-1} - r_{n-1,n}x_n)/r_{n-1,n-1} \ , \\
 & \vdots & & & \vdots & \\
y_n &=& (b_n - \ell_{n,1}y_1 - \cdots - \ell_{n,n-1}y_{n-1})\ell_{n,n}, & x_1 &=& (y_1 - r_{1,2}x_2 - \cdots - r_{1,n}x_n)/r_{1,1} \ .
\end{array}$$

Jeder dieser Prozesse erfordert $\frac{1}{2}n^2 +$ O (n) flops.

Zur Herstellung der LR-Zerlegung verwenden wir die Elementarmatrizen.

$$L_j = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -\ell_{j+1,j} & \ddots & \\ & & \vdots & & \\ & & -\ell_{n,j} & & 1 \end{pmatrix} .$$

Sie weichen nur in der j-ten Spalte unterhalb des Diagonalelementes von der Einheitsmatrix ab und enthalten dort die Elemente $-\ell_{j+1,j}, \ldots, -\ell_{n,j}$. Elementarmatrizen genügen einigen einfachen Rechenregeln.

1) Anwendung auf einen Vektor:

$$L_j \begin{pmatrix} a_1 \\ \vdots \\ a_j \\ a_{j+1} \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} a_1 \\ \vdots \\ a_j \\ a_{j+1} - \ell_{j+1,j} a_j \\ \vdots \\ a_n - \ell_{n,j} a_j \end{pmatrix}$$

2) Anwendung auf eine (n,m)-Matrix A : $L_j A$ entsteht aus A durch Subtraktion des $\ell_{i,j}$-fachen der j-ten Zeile von der i-ten Zeile, $i = j+1, \ldots, n$. Dies ist genau die Operation, die wir beim j-ten Eliminationsschritt durchgeführt haben.

3) Aus 1) folgt sofort: L_j ist invertierbar, und L_j^{-1} entsteht aus L_j durch Streichen der Vorzeichen der $\ell_{i,j}$.

4) Das Produkt $L_j L_k$, $j < k$, berechnet sich durch "Überlagern" von L_j, L_k:

$$L_j L_k = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & -\ell_{j+1,j} & \ddots & & & \\ & & \vdots & & 1 & & \\ & & & & -\ell_{k+1,k} & \ddots & \\ & & -\ell_{n,j} & & -\ell_{n,k} & & 1 \end{pmatrix}$$

Neben den Elementarmatrizen benötigen wir noch Permutationsmatrizen. Sei $\{i_1, \ldots, i_n\}$ eine Permutation von $\{1, \ldots, n\}$ und sei e_i der i-te Einheitsvektor in K^n. Dann ist

$$P = \begin{pmatrix} e_{i_1}^* \\ \vdots \\ e_{i_n}^* \end{pmatrix}$$

die zur Permutation $\{i_1, \ldots, i_n\}$ gehörende Permutationsmatrix. Auch Permutationsmatrizen genügen einfachen Rechenregeln.

1) Anwendung auf einen Vektor:

$$P \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} a_{i_1} \\ \vdots \\ a_{i_n} \end{pmatrix}$$

2) Anwendung auf eine (n, m)-Matrix $A : PA$ entsteht aus A durch Permutation der Zeilen entsprechend der Permutation $\{i_1, \ldots, i_n\}$.

3) AP^* entsteht aus A durch Permutation der Spalten gemäß der Permutation $\{i_1, \ldots, i_n\}$.

4) P ist invertierbar, und $P^{-1} = P^*$. Insbesondere ist $PP^* = P^*P = I$, also P unitär.

Zum Schluß wollen wir noch das Zusammenspiel von Elementar- und Permutationsmatrizen betrachten. Sei P eine Permutationsmatrix, welche nur Zeilen $> j$ vertauscht, d.h. $i_1 = 1, \ldots, i_j = j$. Dann gilt $PL_jP^* = L'_j$, wobei L'_j die gleiche Form wie L_j hat, aber mit (gemäß der Permutation) vertauschten Elementen $\ell'_{k,j} = \ell_{i_k,j}, \;\; k > j$. Dies sieht man sofort, wenn man

$$L_j = I + \begin{pmatrix} 0 & & & & \\ & \ddots & & & \\ & & 0 & & \\ & & -\ell_{j+1,j} & \ddots & \\ & & -\ell_{n,j} & & 0 \end{pmatrix}$$

schreibt und die Wirkung von Links- bzw. Rechtsmultiplikationen mit P und P^* studiert. Es folgt die Vertauschungsrelation $PL_j = L'_jP$.

Wir können nun das Eliminationsverfahren durch rekursive Linksmultiplikation der Matrix (A, b) mit Permutations- und Elementarmatrizen darstellen. Sei P_j die Permutations- und Elementarmatrix, welche die Zeilenvertauschung (z.B. durch maximale Spaltenpivotsuche) vor dem j-ten Eliminationsschritt ausführt. P_j vertauscht also nur die Zeilen $j, \ldots, n$ untereinander. Sei L_j die Elementarmatrix mit $\ell_{i,j} = a_{i,j}/a_{i,j}$, wobei $a_{i,j}$ nun das (i, j)-Element vor Beginn der Elimination im j-ten Eliminationsschritt, also unmittelbar nach Anwendung von P_j ist. Das Eliminationsverfahren lautet dann

$$L_{n-1}P_{n-1} \cdots L_2P_2L_1P_1(A, b) = (R, y)\ .$$

Dabei ist R die rechte Dreiecksmatrix, die im Laufe des Eliminationsprozesses entsteht, und y die zugehörige rechte Seite. Das Produkt auf der linken Seite kann man durch die Vertauschungsregel $P_k L_j = L'_j P_k, \ \ k > j$, vereinfachen. Für $n = 4$ ist z.B.

$$
\begin{aligned}
L_3 P_3 L_2 P_2 L_1 P_1 &= L_3 P_3 L_2 L'_1 P_2 P_1 \\
&= L_3 L'_2 P_3 L'_1 P_2 P_1 \\
&= L_3 L'_2 L''_1 P_3 P_2 P_1 \\
&= L^{-1} P \ \ ,
\end{aligned}
$$

$$
L = (L''_1)^{-1} (L'_2)^{-1} L_3^{-1} \quad , \quad P = P_3 P_2 P_1 \ .
$$

Nach unseren Rechenregeln ist L eine linke Dreiecksmatrix, die durch "Überlagern" der Elemente $\ell_{i,j}$, $i > j$, entsteht. P ist diejenige Permutationsmatrix, die alle während der Elimination aufgetretenen Zeilenvertauschungen ausführt.

Satz 1.2.1 *Zu jeder (n,n)-Matrix A gibt es eine Permutationsmatrix P, so daß PA eine LR-Zerlegung mit $\ell_{i,i} = 1, \ \ i = 1, \ldots, n$ hat.*

Bemerkungen:

1) Für die Gültigkeit von Satz 1 ist es unerheblich, ob A invertierbar ist oder nicht. Ist A nicht invertierbar, so kann man einmal kein von Null verschiedenes Pivotelement mehr finden. In diesem Fall kann man den Eliminationsschritt unterlassen, also $L_j = P_j = I$ setzen.

2) Wie das Beispiel $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ zeigt, ist der Satz ohne die Permutation P falsch.

3) Die LR-Zerlegung von A kann mit Hilfe eines leicht abgeänderten Programms elim hergestellt werden. Die Matrix L erhält man, indem man die ℓ's als $\ell_{i,j}$ speichert. R ist die durch die erste j-Schleife umgeformte Matrix A. P erhält man als Produkt aller ausgeführten Zeilenvertauschungen.

4) Hat man mehrere Systeme $Ax = b^{(i)}$, $i = 1, \ldots, m$ mit ein und derselben Matrix A zu lösen, braucht man die LR-Zerlegung nur einmal auszuführen. Danach kann man jedes System durch $n^2 +$ O (n) flops lösen.

1.3 Die Cholesky-Zerlegung

Eine (n,n)-Matrix $A = (a_{ij})$ heißt hermitesch, wenn $A = A^*$ oder $a_{ij} = \overline{a}_{ji}$. Falls A hermitesch ist, ist $(x, Ay) = (Ax, y)$ reell, und A besitzt n reelle Eigenwerte und ein zugehöriges Orthogonalsystem von n Eigenvektoren. Ist darüber hinaus $(x, Ax) \geq 0$, so heißt A positiv semidefinit. Ist $(Ax, x) > 0$ für $x \neq 0$, so heißt A positiv definit. Ist A positiv definit, so offenbar auch alle Untermatrizen $(a_{ij})_{k \leq i,j \leq \ell}$ mit $k \leq \ell$; insbesondere sind also die Diagonalelemente positiv.

Wir wollen annehmen, die positive definite Matrix A besitze eine LR - Zerlegung. Sei also $A = LR$ und $\ell_{i,i} = 1$, $i = 1, \ldots, n$. Dann ist $A^* = R^* L^*$, also $A = R^* L^*$ eine weitere LR-Zerlegung von A. Sei D die Diagonale von R. Dann ist $A = R^*(D^*)^{-1} D^* L^*$ eine LR-Zerlegung von A, deren linker Faktor nur $1'$sen auf der Hauptdiagonalen hat. Nach der Eindeutigkeit der LR-Zerlegung (Aufgabe 5) ist also $L = R^*(D^*)^{-1}$ und $D^* L^* = R$. Es folgt $D = D^*$, also D reell (im Falle $K = \mathbb{C}$, sonst ist diese Aussage leer). Es ist also $R = DL^*$ mit einer reellen Diagonalmatrix D, d.h. R und L^* stimme bis auf eine reelle Diagonalmatrix überein. Nach geeigneter Fixierung der Diagonalen von L kann man also $R = L^*$ annehmen, und wir kommen zu

$$A = LL^* \; . \tag{3.1}$$

Dies nennt man die Choleskey-Zerlegung von A. Der folgende Satz gibt die genaue Eindeutigkeitsbedingung, sein Beweis einen bequemen Algorithmus.

Satz 1.3.1 *Sei A positiv definit. Dann gibt es genau eine linke Dreiecksmatrix L mit positiven Diagonalelementen, so daß $A = LL^*$.*

Beweis: $A = LL^*$ bedeutet elementweise geschrieben

$$\sum_{k=1}^{j} \ell_{i,k} \overline{\ell}_{j,k} = a_{i,j} \quad , \quad n \geq i \geq j \geq 1 \; ; \tag{3.2}$$

für $K = \mathbb{R}$ kann das „ - " - Zeichen natürlich wegfallen. Das nichtlineare Gleichungssystem (3.2), bestehend aus $n(n+1)/2$ Gleichungen für ebenso viele Unbekannte. Es läßt sich rekursiv auflösen. Die Gleichungen für $j = 1$ lauten

$$\ell_{i,1} \overline{\ell}_{1,1} = a_{i,1} \; , \quad i = 1, \ldots, n \; .$$

Für $i = 1$ ergibt sich $\ell_{1,1} = \sqrt{a_{1,1}}$. Dabei wurde $\ell_{1,1} > 0$ und $a_{1,1} > 0$ benutzt. Für die weiteren Elemente der 1. Spalte von L ergibt sich dann

$$\ell_{i,1} = a_{i,1}/\ell_{1,1} \ , \quad i = 2, \ldots, n \ .$$

Damit ist die erste Spalte von L bestimmt.

Die Gleichungen (3.2) für $j = 2$ lauten

$$\ell_{i,1}\overline{\ell}_{2,1} + \ell_{i,2}\overline{\ell}_{2,2} = a_{i,2} \ , \quad i = 2, \ldots, n \ .$$

Für $j = 2$ ergibt sich

$$\ell_{2,2} = \sqrt{a_{2,2} - |\ell_{2,1}|^2} \ .$$

Dabei wurde angenommen, daß der Radikand positiv ist, und daß $\ell_{2,2} > 0$. Die weiteren Elemente der 2. Spalte von L ergeben sich dann zu

$$\ell_{i,2} = (a_{i,2} - \ell_{i,1}\overline{\ell}_{2,2})/\ell_{2,2} \ , \quad i = 3, \ldots, n \ .$$

Damit ist auch die zweite Spalte von L bestimmt.

Wir wollen nun annehmen, Spalten $1, \ldots, j-1$ von L seien bereits bestimmt. Dann schreiben wir die Gleichung (3.2) für die j-te Spalte hin, also

$$\ell_{i,1}\overline{\ell}_{j,1} + \cdots + \ell_{i,j-1}\overline{\ell}_{j,j-1} + \ell_{i,j}\overline{\ell}_{j,j} = a_{i,j} \ , \quad i = j, \ldots, n \ .$$

Für $i = j$ ergibt sich aus $\ell_{j,j} > 0$ sofort

$$\ell_{j,j} = \sqrt{a_{j,j} - |\ell_{j,1}|^2 - \cdots - |\ell_{j,j-1}|^2} \ , \tag{3.3}$$

wenn nur der Radikand positiv ist. Die weiteren Elemente der j-ten Spalte sind dann

$$\ell_{i,j} = (a_{ij} - \ell_{i,1}\overline{\ell}_{j,1} - \cdots - \ell_{i,j-1}\overline{\ell}_{j,j-1})/\ell_{i,j} \ . \tag{3.4}$$

Die auf der rechten Seite von (3.3), (3.4) auftretenden Elemente von L stehen bis auf $\ell_{j,j}$ alle in den bereits berechneten Spalten von L, und $\ell_{j,j}$ ergibt sich aus (3.3). Wir sehen, daß alle Elemente von L berechnet werden können, wenn nur der Radikand in (3.3) immer positiv ausfällt. Dies wollen wir nun durch vollständige Induktion zeigen. Für $j = 1$ ist dies richtig, weil $a_{1,1} > 0$. Sei es richtig bis zu einem $j \geq 1$. Dann lassen sich die Spalten $1, \ldots, j-1$ von L berechnen. Diese bilden die $(n, j-1)$-Matrix L_j, und es gilt

$$L_j L_j^* = \begin{pmatrix} a_{1,1} & & & & & \\ & \ddots & & & & \\ a_{j-1,1} & \cdots & a_{j-1,j-1} & & & \\ a_{j,1} & \cdots & a_{j,j-1} & x_{jj} & & \\ \vdots & & & & \ddots & \\ a_{n,1} & \cdots & a_{n,j-1} & x_{n,j} & \cdots & x_{n,n} \end{pmatrix} . \tag{3.5}$$

Es wurde nur die linke Hälfte der Matrix notiert; die rechte ergibt sich aus der Hermitezität. Die Elemente $x_{i,j}$ sind ohne Bedeutung mit Ausnahme von $x_{j,j}$, und dieses ist

$$x_{j,j} = |\ell_{j,1}|^2 + \cdots + |\ell_{j,j-1}|^2 .$$

Wir zeigen, daß $x_{j,j} < a_{j,j}$. Wäre nämlich $x_{j,j} \geq a_{j,j}$, so wäre die (j,j)-Matrix

$$\begin{pmatrix} a_{1,1} & & \\ \vdots & \ddots & \\ a_{j-1,1} & a_{j-1,j-1} & \\ a_{j,1} & a_{j-1,j} & x_{j,j} \end{pmatrix}$$

(wieder haben wir nur die linke Hälfte notiert) positiv definit, denn dies wäre ja schon für $x_{j,j} = a_{j,j}$ richtig, um so mehr also für $x_{j,j} > a_{j,j}$. Damit hätte aber die rechte Seite von (3.5) mindestens den Rang j, während die linke Seite als Produkt von Matrizen mit höchstens $j-1$ Zeilen bzw. Spalten höchstens den Rang $j-1$ haben kann. Es kann also $x_{j,j} \geq a_{j,j}$ nicht richtig sein. Damit ist $x_{j,j} < a_{j,j}$ für $j = 1, \ldots, n$, d.h. der Radikant in (3.3) ist immer positiv, und L läßt sich in eindeutiger Weise bestimmen.

Der Beweis führt sofort zu einem Programm für die Cholesky-Zerlegung.

Cholesky (A, n) $/\star$ Überschreibt den links unteren Teil von A mit seiner Cholesky-Zerlegung L. Arbeitet nur auf dem linken unteren Teil von A $\star/$

```
{ for  j = 1,...,n
        for  i = j,...,n
          {  s = a_{i,j} - a_{i,1} * ā_{j,1} - ··· - a_{i,j-1} * ā_{j,j-1} ;
            if (i = j)
               { if (s ≤ 0) { print ("Matrix nicht pos. def.") ;
                                exit (1) ;
                            }
                 else a_{j,j} = sqrt (s) ;
               }
            else a_{i,j} = s/a_{j,j} ;
          }
}
```

Für die Anzahl der benötigten flops findet man

$$\sum_{j=1}^{n} (n-j)(j-1+\mathrm{O}(1)) = \frac{1}{6}n^3 + \mathrm{O}(n^2)\,.$$

Dies entspricht dem Eliminationsverfahren für hermitesche Matrizen (vergleiche Übungsaufgabe 7). Das Gleichungssystem $Ax = b$ kann nach Berechnung von L gelöst werden durch Lösung der Systeme $Ly = b$, $L^*x = y$ wie nach der LR-Zerlegung.

1.4 Die QR-Zerlegung

Sei A eine (n,m)-Matrix, $n \geq m$, mit linear unabhängigen Spalten $a_1, \ldots, a_m \in K^n$. Bekanntlich kann man die Spalten von A orthogonalisieren, d.h. man kann ein Orthonormalsystem von Vektoren $q_1, \ldots, q_m$ finden, so daß

$$\mathrm{sp}(a_1, \ldots, a_j) = \mathrm{sp}(q_1, \ldots, q_j) \ , \quad j = 1, \ldots, m \tag{4.1}$$

gilt. Hierbei bedeutet $\mathrm{sp}(a, b, \ldots)$ den von $a, b, \ldots$ aufgespannten linearen Unterraum. Die Orthogonalisierung kann durch das Schmidtsche Verfahren gemacht werden. Wir setzen

$$q_1 = a_1 / r_{1,1} \ , \quad r_{1,1} = \|a_1\|$$

mit der euklidischen Norm. Damit ist (4.1) für $j = 1$ erfüllt. Haben wir bereits orthonormale Vektoren $q_1, \ldots, q_{j-1}$ bestimmt, so setzen wir

$$\hat{q}_j = a_j - r_{1,j} q_1 - \cdots - r_{j-1,j} q_{j-1} \tag{4.2}$$

und bestimmen die $r_{i,j}$, $i = 1, \ldots, j-1$, so, daß $(\hat{q}_j, q_i) = 0$, $i = 1, \ldots, j-1$, also

$$r_{i,j} = (a_j, q_i) \ . \tag{4.3}$$

Dann setzen wir

$$q_j = \hat{q}_j / r_{j,j} \quad , \quad r_{j,j} = \|\hat{q}_j\| \ .$$

So bestimmen wir rekursiv $q_1, \ldots, q_m$. Die $r_{j,j}$ können nicht verschwinden, wenn $a_1, \ldots, a_m$ linear unabhängig sind. Offenbar gilt

$$a_j = r_{1,j} q_1 + \cdots + r_{j,j} q_j \ , \quad j = 1, \ldots, m \ .$$

Mit der (n,n)-Matrix $Q = (q_1, \ldots, q_m)$ und der rechten (m,m)-Dreiecksmatrix R, deren (i,j)-Element $r_{i,j}$ ist für $i \leq j$ lautet dies

$$A = QR \ . \tag{4.4}$$

Dies ist die QR-Zerlegung von A. Nach Herleitung ist diese Zerlegung in eine orthonormale Matrix Q und eine rechte Dreiecksmatrix R eindeutig bestimmt, wenn man die Diagonalelemente von R positiv annimmt.

Numerisch ist das Schmidtsche Verfahren völlig ungeeignet. Nach (4.2), (4.3) ist nämlich

$$\|\hat{q}_j\| = \min_{q \in \mathrm{sp}(q_1,\ldots,q_{j-1})} \|a_j - q\| \,.$$

$\hat{q}_j$ wird also sehr klein ausfallen, insbesondere dann, wenn a_j fast linear abhängig von $a_1, \ldots, a_{j-1}$ ist. Bei der Berechnung von $\hat{q}_j$ treten also Auslöschungen auf, so daß auch q_j nicht genau berechnet werden kann.

Ein stabiles Verfahren zur QR-Zerlegung ist das Householder-Verfahren. Wir beschreiben es für reelle Matrizen A. Es macht Gebrauch von Spiegelungen

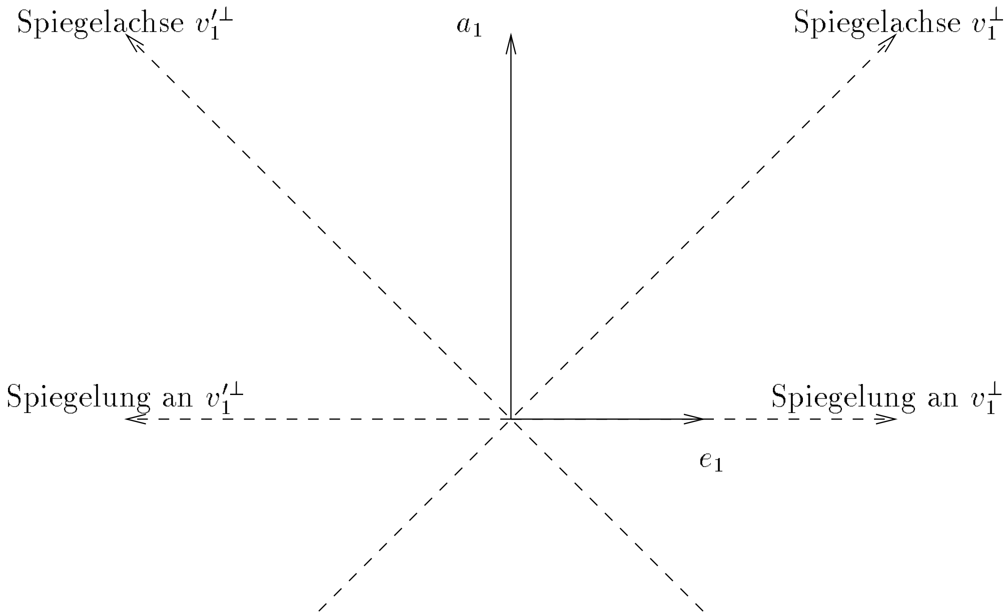
$$S = I - 2vv^* \quad , \quad \|v\| = 1$$

an der Hyperebene $v^\perp$. Dabei tritt das dyadische Produkt $(vv^*)_{ij} = v_i v_j$ auf. Es ist

$$\begin{aligned} S^2 &= (I - 2vv^*)(I - 2vv^*) &&= I - 4vv^* + 4vv^*vv^* \\ &= I - 4vv^* + 4vv^* &&= I \end{aligned}$$

und $S = S^*$. Also ist $SS^* = S^*S = I$, d.h. S unitär.

Das Householder - Verfahren bestimmt Spiegelungen $S_1, \ldots, S_{m-1}$, so daß $S_j \cdots S_1 A$ in den Spalten $1, \ldots, j$ bereits rechte Dreiecksgestalt hat. Wir beschreiben ausführlich die Bestimmung von S_1. Die erste Spalte von $S_1 A$ lautet $S_1 a_1$. Wir müssen S_1 also so bestimmen, daß $S_1 a_1$ ein Vielfaches von e_1 ist. Dies kann man auf zwei Weisen erreichen, nämlich



durch Spiegelung an $v_1^{\perp}$, wo $v_1 = (a_1 + \alpha_1 e_1)/\beta_1$, $\beta_1 = \|a_1 + \alpha_1 e_1\|$ mit $\alpha_1 = \pm\|a_1\|$. Zur Vermeidung von Auslöschung wählt man
$\alpha_1 = \|a_1\| \text{ sgn } (a_{1,1})$. Mit

$$\begin{aligned} \beta_1^2 &= \|a_1\|^2 + \alpha_1^2 + 2\alpha a_{1,1} &&= 2\alpha_1(\alpha_1 + a_{1,1}) \\ 2v_1^* a_1 &= \tfrac{2}{\beta_1}(a_1 + \alpha_1 e_1)^* a_1 &&= \tfrac{2}{\beta_1}(\|a_1\|^2 + \alpha_1 a_{1,1}) \\ &= \tfrac{2}{\beta_1}\alpha_1(\alpha_1 + a_{1,1}) &&= \beta_1 \end{aligned}$$

erhalten wir, wie erwartet, für die erste Spalte von $S_1 A$

$$\begin{aligned} S_1 a_1 &= (I - 2v_1 v_1^*) a_1 = a_1 - 2v_1 v_1^* a_1 \\ &= a_1 - \beta_1 v_1 = a_1 - (a_1 + \alpha_1 e_1) = -\alpha_1 e_1 \ . \end{aligned}$$

Die weiteren Spalten sind

$$S_1 a_k = a_k - 2v_1 v_1^* a_k \quad , \quad k = 2, \ldots, m \ .$$

$S_1 A$ hat die Gestalt

$$S_1 A = \begin{pmatrix} -\alpha_1 & r_{1,2} & \cdots & r_{1,m} \\ 0 & & & \\ \vdots & & A_2 & \\ 0 & & & \end{pmatrix} ,$$

wobei A_2 eine $(n-1, n-1)$-Matrix ist. Wir eliminieren nun die Elemente unterhalb des $(2,2)$-Elements durch Linksmultiplikation mit einer Spiegelung der Form

$$S_2 = \begin{pmatrix} 1 & 0 & \cdots & & 0 \\ 0 & & & & \\ \vdots & & I - 2v_2v_2^* & & \\ 0 & & & & \end{pmatrix},$$

wo nun I die $(n-1, n-1)$ Einheitsmatrix und $v_2 \in \mathbb{R}^{n-1}$, $\|v_2\| = 1$ bedeuten. v_2 berechnet sich aus der ersten Spalte von A_2 genau so, wie sich v_1 aus a_1 berechnete. Es wird dann

$$S_2S_1A = \begin{pmatrix} -\alpha_1 & r_{1,2} & \cdots & & r_{1,m} \\ 0 & -\alpha_2 & r_{2,3} & \cdots & r_{2,m} \\ \vdots & 0 & & & \\ & \vdots & & A_3 & \\ 0 & 0 & & & \end{pmatrix}$$

mit einer $(n-2, n-2)$-Matrix A_3. Nach m Schritten erhält man

$$S_m \cdots S_1 A = \begin{pmatrix} R \\ 0 \end{pmatrix}$$

mit der rechten Dreiecksmatrix

$$R = \begin{pmatrix} -\alpha_1 & r_{1,2} & \cdots & r_{1,m} \\ & \ddots & & \vdots \\ \mathbf{O} & & & r_{m-1,m} \\ & & & -\alpha_m \end{pmatrix}.$$

Es folgt

$$A = S_1 \cdots S_m \begin{pmatrix} R \\ 0 \end{pmatrix} = S \begin{pmatrix} R \\ 0 \end{pmatrix} = QR,$$

wobei Q aus den Spalten $1, \ldots, m$ von S besteht. Damit haben wir die QR-Zerlegung von A gefunden.

Es ist nicht zweckmäßig, die Matrix S oder Q wirklich zu berechnen. Es ist nämlich sehr einfach, Sx alleine mit Hilfe der Vektoren $v_j \in \mathbb{R}^{n-j+1}$ zu berechnen. Es ist nämlich

$$S_j x = \begin{pmatrix} x^{j-1} \\ x^{n-j+1} - 2(v_j, x^{n-j+1})v_j \end{pmatrix} \quad , \quad x = \begin{pmatrix} x^{j-1} \\ x^{n-j+1} \end{pmatrix} \ ,$$

wobei x^{j-1}, x^{n-j+1} die Längen $j-1$ bzw. $n-j+1$ haben.

Dies verlangt nur $2(n-j+1)+\mathrm{O}(1)$ flops. Die sukzessive Berechnung von $Sx = S_1 \cdots S_m x$ erfordert daher etwa für $n = m$ nur $m^2 + \mathrm{O}(m)$ flops, also ebenso viele wie die Berechnung von Sx bei vorberechnetem S.

Ein Programm zur QR-Zerlegung berechnet also zweckmäßigerweise nicht Q, sondern die Spiegelungsvektoren $v_1, \ldots, v_m$.

$qr\_$ dcmp (A, α, n, m)

/ $\star$ Führt die QR-Zerlegung der (n, m)-Matrix A durch $(n \geq m)$. Nach Ablauf enthält A in und unterhalb der Diagonalen die Vektoren $v_1, \ldots, v_m$ und oberhalb der Diagonalen die Außerdiagonalelemente von R. Die (negativen) Diagonalelemente von R werden auf den Vektor α geschrieben. $\star$ /

$$
\begin{array}{lll}
\{ & \text{for} & j = 1, \ldots, m \\
 & \{ & \alpha_j = \|a_j\| * \operatorname{sgn}\ (a_{j,j});\ \beta = \sqrt{2\alpha_j * (\alpha_j + a_{j,j})}; \\
 & & a_{j,j} = (a_{j,j} + \alpha_j)/\beta; \\
 & & \text{for } i = j+1, \ldots, n \ \ a_{i,j} = a_{i,j}/\beta; \\
 & & \text{for } k = j+1, \ldots, m \\
 & & \{\ \gamma = 2 * (a_{j,k} * a_{j,j} + \cdots + a_{n,k} * a_{n,j}); \\
 & & \quad \text{for } i = j, \ldots, n \quad a_{i,k} = a_{i,k} - \gamma * a_{i,j}; \\
 & & \} \\
 & \} & \\
\} & &
\end{array}
$$

Dieses Programm wird noch ergänzt durch zwei weitere Programme, welche Sx bzw. S^*x bilden.

q_{-} mal_$x(A, x, n, m)$

```
/ ⋆   Überschreibt x mit Sx nach Aufruf von
      qr_ dcmp (A, α, n, m) ⋆ /
{     for j = m, ..., 1
      { γ = 2 * (a_{j,j} * x_j + ··· + a_{n,j} * x_n);
        for i = j, ..., n  x_i = x_i − γ * a_{i,j};
      }
}
```

q_{-}^{*} mal_$x(A, x, n, m)$

```
/ ⋆   Überschreibt x mit S*x nach Aufruf von
      qr_ dcmp (A, α, n, m) ⋆ /
{       Wie oben, aber  j = 1, ..., m}
```

1.5 Unter- und überbestimmte lineare Systeme

Sei A eine (n, m)-Matrix über K. Das lineare Gleichungssystem $Ax = b$ heißt überbestimmt für $n > m$, unterbestimmt für $n < m$. Im ersten Fall ist $Ax = b$ in der Regel unlösbar, im zweiten Fall in der Regel nicht eindeutig lösbar. Wir wollen eine verallgemeinerte Lösung von $Ax = b$ definieren, welche immer eindeutig bestimmt ist, und Verfahren zu deren Berechnung angeben.

Wir bezeichnen mit $ker(A)$ den Nullraum, mit $range(A)$ den Wertebereich von A, also

$$\begin{array}{lcl} ker(A) & = & \{x \in K^m : Ax = 0\} \ , \\ range(A) & = & \{y \in K^n : \exists \ x \in K^m \ \text{ mit } \ y = Ax\} \ . \end{array}$$

Weiter bezeichnen wir für lineare Unterräume U, V von K^n mit $U + V$ die Summen von Vektoren aus U, V. Ist $U \perp V$, so schreiben wir für $U + V$ auch $U \oplus V$. Für später notieren wir, daß $ker(A) = ker(A^*A)$.

Aus der linearen Algebra erinnert man, daß $Ax = b$ genau dann lösbar ist, wenn $b \perp ker(A^*)$. Wir wollen dies etwas anders formulieren.

Satz 1.5.1 *Für jede (n, m)-Matrix A gilt*

$$K^n = ker(A^*) \ \oplus \ range(A) \ .$$

Beweis: Zunächst ist $ker(A^*) \perp range(A)$. Ist nämlich $A^*x = 0$ und $y = Az$, so folgt

$$(x, y) = (x, Az) = (A^*x, z) = 0 \ ,$$

also $x \perp y$. Es bleibt zu zeigen, daß $ker(A^*) + range(A) = K^n$ ist. Wäre dies nicht der Fall, so gäbe es ein $y \in K^n$ mit $y \neq 0$ und $y \perp ker(A^*)$, $y \perp range(A)$. Wegen $y \perp ker(A^*)$ wäre $Ax = y$ lösbar, also $y \in range(A)$. Dies steht im Widerspruch zu $y \perp range(A)$ und $y \neq 0$.

□

Als ersten Schritt zur Definition einer verallgemeinerten Lösung schwächen wir den Lösungsbegriff ab. Wir verlangen nicht mehr, daß $Ax - b = 0$ ist, sondern nur noch, daß $\|Ax - b\|$ möglichst klein ist. Dabei verwenden wir die euklidische Norm.

Satz 1.5.2 *$\|Ax-b\|$ nimmt genau dann für $x = x_0$ sein Minimum an, wenn $A^*Ax_0 = A^*b$.*

Beweis: Nach Satz 1 ist $b = b_1 + b_2$ mit $b_1 \in range(A)$ und $b_2 \in ker(A^*)$. Dann ist $Ax - b_1 \perp b_2$, und wir haben

$$\|Ax - b\|^2 = \|Ax - b_1 - b_2\|^2 = \|Ax - b_1\|^2 + \|b_2\|^2 \ .$$

$\|Ax - b\|$ ist also minimal genau dann, wenn $Ax - b_1 = 0$, und dies ist genau dann der Fall, wenn $A^*(Ax - b) = 0$.

□

Bemerkungen:

1) $A^*Ax = A^*b$ heißt System der Normalgleichungen. x^0 heißt Kleinste-Quadrate-Lösung. Man spricht auch von der (Gaußschen) Methode der kleinsten Quadrate.

2) Die Normalgleichungen sind immer lösbar. Dies folgt natürlich aus dem eben bewiesenen Satz. Man kann es aber auch direkt bestätigen. Es ist ja $ker(A) = ker(A^*A)$, also

$$A^*b \in range(A^*) \ \perp \ ker(A)$$

und damit $A^*b \perp ker(A^*A)$.

Die zweite Bemerkung zeigt, daß die Kleinste-Quadrate-Lösung von $Ax = b$ immer existiert. Leider ist sie i. allg. nicht eindeutig.

Definition 1.5.1 *x^+ heißt verallgemeinerte (Moore-Penrose-) Lösung von $Ax = b$, wenn*

1) x^+ ist Kleinste-Quadrate-Lösung.

2) Unter allen Kleinste-Quadrate-Lösungen hat x^+ minimale Norm.

Satz 1.5.3 *x^+ existiert und ist eindeutig bestimmt. x^+ ist genau dann verallgemeinerte Lösung, wenn*

$$A^*Ax^+ = A^*b\,, \qquad x^+ \in range(A^*)\;.$$

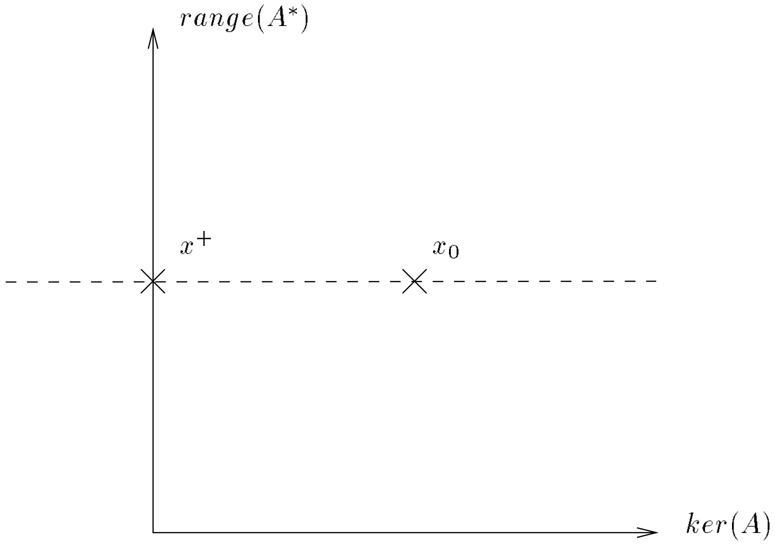
Beweis: Sei x_0 Kleinste-Quadrate-Lösung, also $A^*Ax_0 = A^*b$. Nach Satz 1, angewandt auf A^*, ist $x_0 = x_1 + x_2$ mit $x_1 \in range(A^*)$, $x_2 \in ker(A)$. Auch x_1 erfüllt die Normalgleichungen und ist damit Kleinste-Quadrate-Lösung. Es gibt also immer eine Kleinste-Quadrate-Lösung x_1 in $range(A^*)$. Jede weitere Kleinste-Quadrate-Lösung x ist dann von der Gestalt $x = x_1 + y$ mit $y \in ker(A^*A) = ker(A)$. Wegen $x_1 \perp y$ ist

$$\|x\|^2 = \|x_1\|^2 + \|y\|^2\;.$$

Die Kleinste-Quadrate-Lösung x^+ minimaler Norm erhält man also in eindeutiger Weise durch $y = 0$ oder $x^+ = x_1$.

□

Für $m = 2$ und $K = \mathbb{R}$ wird Satz 3 auch aus folgender Zeichnung klar:



Die gestrichelte Linie ist der affine Unterraum der Kleinste-Quadrate-Lösungen.

Die Zuordnung $b \to x^+$ ist offenbar linear. Also gibt es eine (m,n)-Matrix A^+ mit $x^+ = A^+b$. A^+ heißt verallgemeinerte (Moore-Penrose-) Inverse von A. Ist $n = m$ und A invertierbar, so ist natürlich $A^+ = A^{-1}$. Hat A vollen Rang, so kann man A^+ aus Satz 3 leicht berechnen. Für $n \geq m$ sind dann nämlich die Normalgleichungen eindeutig lösbar, und man bekommt sofort

$$A^+ = (A^*A)^{-1}A^* .$$

Ist $n < m$, so ist $x^+ = A^*y$ und $A^*AA^*y = A^*b$, also $AA^*y = b$. Da jetzt AA^* invertierbar ist, folgt $y = (AA^*)^{-1}b$ und $x^+ = A^*(AA^*)^{-1}b$. Also ist in diesem Fall

$$A^+ = A^*(AA^*)^{-1} .$$

Die Standard-Methode zur Berechnung der verallgemeinerten Lösung von $Ax = b$ ist die QR-Zerlegung. A besitze vollen Rang. Im überbestimmten Fall, also $n \geq m$, führen wir zunächst die QR-Zerlegung von A durch. Die Normalgleichungen lauten dann

$$A^*Ax = R^*Q^*QRx^+ = R^*Rx^+ = R^*Q^*b$$

oder, da R vollen Rang hat,

$$Rx^+ = Q^*b . \tag{5.1}$$

x^+ berechnet sich nun durch Rückwärtseinsetzen. Für A^+ bedeutet dies $A^+ = R^{-1}Q^*$.

Im unterbestimmten Fall, also $n \leq m$, beginnen wir mit der QR-Zerlegung von A^*. Die x^+ charakterisierenden Gleichungen

$$A^*Ax^+ = A^*b , \qquad x^+ = A^*y$$

schreiben sich dann als

$$QRR^*Q^*Qz = QRb , \quad x^+ = Qz .$$

Es ist $Q^*Q = I$, und QR hat maximalen Rang. Also folgt

$$R^*z = b , \qquad x^+ = Qz . \tag{5.2}$$

Jetzt kann z durch Vorwärtseinsetzen berechnet werden. A^+ ergibt sich zu $A^+ = Q(R^*)^{-1}$.

1.6 Fehlerabschätzung bei linearen Gleichungssystemen

Im letzten Paragraphen haben wir eine Methode zur Bestimmung der Lösung eines linearen Gleichungssystems kennengelernt. Wir werden nun die Abhängigkeit dieser Lösung von Störungen untersuchen. Hierzu zunächst ein

Beispiel:

Löse $Ax = b$ mit

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 0.99 \end{pmatrix} \quad , \quad b = \begin{pmatrix} 1 \\ 1 \end{pmatrix} .$$

Wir erhalten die Lösung $x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. Statt A, b seien nun nur die fehlerbehafteten Näherungen $\tilde{A}$, $\tilde{b}$ bekannt:

$$\tilde{A} = \begin{pmatrix} 1.01 & 1.01 \\ 1 & 0.99 \end{pmatrix} \quad , \quad \tilde{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} .$$

Die Lösung $\tilde{x}$ von $\tilde{A}\tilde{x} = \tilde{b}$ ist

$$\tilde{x} = \begin{pmatrix} 200/101 \\ -100/101 \end{pmatrix} .$$

Obwohl wir also einen Fehler von nur 1% in den Daten haben, bekommen wir einen Fehler von über 100% in der Lösung.

Wir werden versuchen, dieses Phänomen zu erklären. Hierzu wiederholen wir einige Grundbegriffe der linearen Algebra.

Definition 1.6.1 *Eine Abbildung* $\| \cdot \| : V \to \mathbb{R}^{\geq 0}$ *eines* $\mathbb{C}$*-Vektorraums* V *in die nichtnegativen reellen Zahlen heißt Norm, falls für alle* $x, y \in V$, $\alpha \in \mathbb{C}$ *gilt*

1) $\|x\| = 0 \Leftrightarrow x = 0$

2) $\|\alpha x\| = |\alpha| \|x\|$

3) $\|x+y\| \leq \|x\| + \|y\|$ *(Dreiecksungleichung).*

Beispiele:

Sei $V = \mathbb{C}^n$. Wir benutzen

a) Euklidische Norm: $\|x\|_2 = \left(\sum\limits_{k=1}^{n} |x_k|^2\right)^{1/2}$

b) ∞-Norm: $\|x\|_\infty := \max_k |x_k|$

c) T-Norm: $\|x\|_T := \|Tx\|_\infty$

Wir zeigen für $\|\cdot\|_T$ die Dreiecksungleichung:

$$\|x+y\|_T = \|Tx+Ty\|_\infty \leq \|Tx\|_\infty + \|Ty\|_\infty = \|x\|_T + \|y\|_T$$

Definition 1.6.2 $\|\cdot\|_1$, $\|\cdot\|_2$ *heißen äquivalent, falls es positive Konstanten* c_1, c_2 *gibt mit*

$$c_1\|\cdot\|_1 \;\leq\; \|\cdot\|_2 \;\leq\; c_2\|\cdot\|_1$$

Satz 1.6.1 *In endlich-dimensionalen Vektorräumen sind alle Normen äquivalent.*

Beweis: In jedem der angegebenen Lehrbücher.

Korollar 1.6.1 *Sei* (x_n) *eine Folge im Vektorraum* V *und* V *endlich-dimensional. Seien* $||\cdot||_1$ *und* $||\cdot||_2$ *Normen in* V*. Dann gilt:*

$$\begin{array}{rl} & x_n \textit{ konvergiert gegen } x \quad \textit{bezüglich Norm } \|\cdot\|_1 \\ \Leftrightarrow & x_n \textit{ konvergiert gegen } x \quad \textit{bezüglich Norm } \|\cdot\|_2. \end{array}$$

Wir wollen nun spezielle Normen im Vektorraum der Matrizen definieren.

Definition 1.6.3 *Sei $\| \cdot \|$ eine Norm in $\mathbb{C}^n$. Dann heißt*

$$\| \cdot \| : \mathbb{C}^{(n,n)} \to \mathbb{R}^{\geq 0} , \qquad \|A\| := \sup_{x \in \mathbb{C}^n} \frac{\|Ax\|}{\|x\|}$$

die zugeordnete Matrizennorm.

Bemerkung: Es gilt

$$\|A\| = \sup_{x \in \mathbb{C}^n} \frac{\|Ax\|}{\|x\|} = \sup_{x \in \mathbb{C}^n} \|A \frac{x}{\|x\|}\| = \sup_{\|x\|=1,\, x \in \mathbb{C}^n} \|Ax\| .$$

Korollar 1.6.2 *Die zugeordnete Matrizennorm $\| \cdot \|$ hat folgende Eigenschaften:*

1) $\| \cdot \|$ ist eine Norm im Vektorraum der Matrizen.

2) Sei λ Eigenvektor von A. Dann gilt $\|A\| \geq |\lambda|$.
Beweis: Sei x Eigenvektor von A zum Eigenwert λ, $\|x\| = 1$.
Dann ist $\|A\| \geq \|Ax\| = \|\lambda x\| = |\lambda| \|x\| = |\lambda|$.

3) $\|AB\| \leq \|A\| \|B\|$.
Beweis: Sei $B \neq 0$.

$$\|AB\| = \sup_{x \in \mathbb{C}^n} \frac{\|ABx\|}{\|x\|} = \sup_{x \in \mathbb{C}^n,\, Bx \neq 0} \frac{\|ABx\|}{\|Bx\|} \cdot \frac{\|Bx\|}{\|x\|} \leq \|A\| \cdot \|B\|.$$

Beispiele:

a) Unendlichnorm: Sei $\|x\|_\infty = 1,\ A \neq 0$.

$$\begin{aligned} \|Ax\|_\infty &= \max_i |\sum_{k=1}^n a_{ik} x_k| \\ &\leq \max_i \sum_{k=1}^n |a_{ik}| |x_k| \\ &\leq \max_i \sum_{k=1}^n |a_{ik}|, \quad \text{also} \quad \|Ax\|_\infty \leq \max_i \sum_{k=1}^n |a_{ik}| . \end{aligned}$$

Das $\max\limits_i \sum\limits_{k=1}^{n} |a_{ik}|$ werde für $i = j$ angenommen. Definiere

$$\tilde{x}_k := \begin{cases} \overline{a_{jk}}/|a_{jk}| & , \quad a_{jk} \neq 0 \\ 0 & \quad \text{sonst} . \end{cases}$$

Dann gilt $\|\tilde{x}\| = 1$, und es ist

$$\|A\tilde{x}\|_\infty = \max_i |\sum_{k=1}^{n} a_{ik}x_k| \geq |\sum_{k=1}^{n} a_{jk}x_k| = \sum_{k=1}^{n} |a_{jk}| = \max_i \sum_{k=1}^{n} |a_{ik}| .$$

Also gilt $\|A\| = \max_i \sum\limits_{k=1}^{n} |a_{ik}|$. Für $A = 0$ ist dies klar.

b) Euklidische Norm: Es gilt $\|A\|_2 = \rho(A^*A)^{1/2}$, wobei $\rho(X)$ der Betrag des betragsmäßig größten Eigenwerts von X ist.
Beweis: Übungsaufgabe 17.

c) T–Norm:

$$\begin{aligned} \| \cdot \|_T : \|A\|_T &= \sup_{x \in \mathbb{C}^n} \frac{\|TAx\|_\infty}{\|Tx\|_\infty} = \sup_{Ty \in \mathbb{C}^n} \frac{\|TAT^{-1}y\|_\infty}{\|TT^{-1}y\|_\infty} \\ &= \sup_{y \in \mathbb{C}^n} \frac{\|TAT^{-1}y\|_\infty}{\|y\|_\infty} = \|TAT^{-1}\|_\infty . \end{aligned}$$

Wir haben gezeigt, daß $\rho(A) \leq \|A\|$ für jede Matrix A und daß für hermitesche Matrizen $\|A\|_2 = (\rho(A^*A))^{1/2} = \rho(A^2)^{1/2} = \rho(A)$. Man kann daher fragen: Gibt es für jede Matrix A eine (von A abhängige) Vektornorm $\| \cdot \|^A$, so daß $\|A\|^A = \rho(A)$? Die Antwort ist nein, aber es gilt der

Satz 1.6.2 *Zu jedem $A \in \mathbb{C}^{(n,n)}$ und jedem $\varepsilon > 0$ existiert eine Norm $\| \cdot \|_{A,\varepsilon}$ auf dem $\mathbb{C}^n$, so daß für die zugeordnete Matrixnorm*

$$\|A\|_{A,\varepsilon} \leq \rho(A) + \varepsilon .$$

gilt.

Beweis: Sei

$$D = \operatorname{diag}(1, \varepsilon, \ldots, \varepsilon^{n-1}) = \begin{pmatrix} 1 & & & \mathbf{O} \\ & \varepsilon & & \\ & & \ddots & \\ \mathbf{O} & & & \varepsilon^{n-1} \end{pmatrix} .$$

Bei der Bildung von BD für eine Matrix B wird Spalte k von B mit ε^{k-1} multipliziert. Bei der Bildung von $D^{-1}B$ wird Zeile k von B mit ε^{-k+1} multipliziert.
Sei nun $J = P^{-1}AP$ die Jordansche Normalform von A. J hat die Form

$$\begin{pmatrix} \lambda_1 & \mu_1 & & \mathbf{O} \\ & \ddots & \ddots & \\ & & \ddots & \mu_{n-1} \\ \mathbf{O} & & & \lambda_n \end{pmatrix} ,$$

wobei die λ_k Eigenwerte von A sind und $\mu_k \in \{1, 0\}$. Dann hat $C := D^{-1}JD$ die Form

$$\begin{pmatrix} \lambda_1 & \varepsilon\mu_1 & & \mathbf{O} \\ & \ddots & \ddots & \\ & & \ddots & \varepsilon\mu_{n-1} \\ \mathbf{O} & & & \lambda_n \end{pmatrix} .$$

Definiere $\|x\|_{A,\varepsilon} := \|x\|_T$ mit $T := (PD)^{-1}$. Dann gilt:

$$\|A\|_T = \|D^{-1}P^{-1}APD\|_\infty = \|C\|_\infty \leq \rho(A) + \varepsilon .$$

□

Wir haben damit alle benötigten Hilfsmittel zur Verfügung gestellt. Zur Untersuchung der Fehlerabhängigkeit benutzen wir

Definition 1.6.4 *Sei $A \in \mathbb{C}^{(n,n)}$ invertierbar. Dann heißt $k(A) = \|A\|\|A^{-1}\|$ die Kondition von A bezüglich $\|\cdot\|$.*

Satz 1.6.3 *Sei $A \in \mathbb{C}^{n\times n}$ invertierbar, und $\Delta A \in \mathbb{C}^{(n,n)}$ eine Matrix mit $\|A^{-1}\|\|\Delta A\| < 1$. Dann gilt:*

a) *$(A + \Delta A)$ ist invertierbar,*

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

b) *Sei $b \in \mathbb{C}^n \backslash \{0\}, \Delta b \in \mathbb{C}^n$. Seien x, Δx die Lösungen von $Ax = b$ und $(A + \Delta A)(x + \Delta x) = (b + \Delta b)$. Dann gilt*

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{k(A)}{1-k(A)\frac{\|\Delta A\|}{\|A\|}}\{\frac{\|\Delta A\|}{\|A\|}+\frac{\|\Delta b\|}{\|b\|}\}\ .$$

Beweis:

zu a) Es gilt für $y \neq 0$

$$\|(I + A^{-1}\Delta A)y\| \geq \|y\| - \|A^{-1}\Delta Ay\| \geq \|y\|(1-\|A^{-1}\|\|\Delta A\|) > 0.$$

Die zur Matrix $(I + A^{-1}\Delta A)$ gehörende lineare Abbildung ist also injektiv und damit ein Vektorraumisomorphismus als Abbildung von $\mathbb{C}^n$ nach $\mathbb{C}^n$. Deshalb sind $(I + A^{-1}\Delta A)$ und $A + \Delta A = A(I + A^{-1}\Delta A)$ invertierbar.

Setze nun $y = (I + A^{-1}\Delta A)^{-1}x$. Durch Einsetzen erhalten wir

$$\|x\| \geq \|(I + A^{-1}\Delta A)^{-1}x\|(1-\|A^{-1}\|\|\Delta A\|)\ .$$

Für jedes x gilt damit

$$\|(I + A^{-1}\Delta A)^{-1}x\| \leq \frac{1}{1-\|A^{-1}\|\|\Delta A\|}\cdot\|x\|$$

oder

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

Damit gilt

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

zu b) Wir betrachten zunächst die Gleichungssysteme

$$\begin{aligned} (A+\Delta A)(x+\Delta x) &= b+\Delta b, \\ Ax &= b. \end{aligned}$$

Durch Subtraktion erhalten wir

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

und damit

$$\Delta x = (A+\Delta A)^{-1}(\Delta b - \Delta A x).$$

Mit den Norm–Rechenregeln gilt

$$\begin{aligned} \frac{\|\Delta x\|}{\|x\|} &\leq \frac{1}{\|x\|}\|(A+\Delta A)^{-1}\|\|\Delta b - \Delta A x\| \\ &\leq \frac{\|A^{-1}\|}{1-\|A^{-1}\|\|\Delta A\|}\left(\frac{\|\Delta b\|}{\|x\|}+\|\Delta A\|\right) \\ &= \frac{k(A)}{1-k(A)\frac{\|\Delta A\|}{\|A\|}}\left(\frac{\|\Delta b\|}{\|A\|\|x\|}+\frac{\|\Delta A\|}{\|A\|}\right) \\ &\leq \frac{k(A)}{1-k(A)\frac{\|\Delta A\|}{\|A\|}}\left(\frac{\|\Delta b\|}{\|b\|}+\frac{\|\Delta A\|}{\|A\|}\right) . \end{aligned}$$

$\square$

Wir können den Satz so interpretieren:

Bei kleinem Fehler $\|\Delta A\|$ wird der relative Fehler von Matrix und Ergebnisvektor um den Faktor $k(A)$ erhöht (verstärkt).

Wir wollen nun unser einführendes Beispiel aufklären. Es galt:

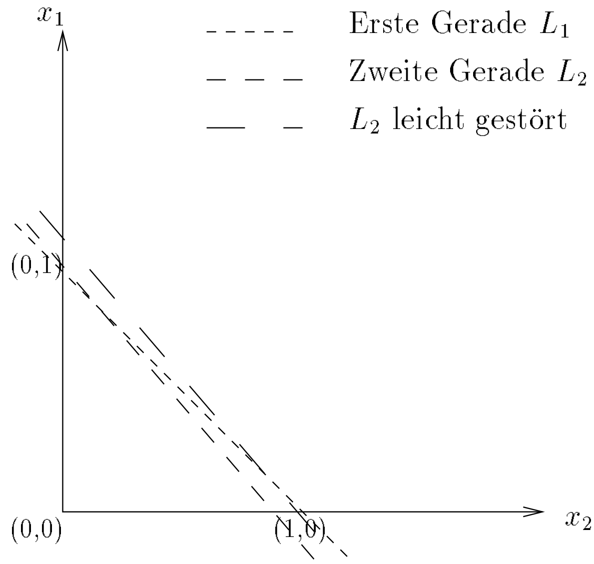
$$A = \begin{pmatrix} 1 & 1 \\ 1 & 0.99 \end{pmatrix} , \quad A^{-1} = \begin{pmatrix} 99 & -100 \\ -100 & 100 \end{pmatrix} .$$

Wir erhalten:

$$k(A)_\infty = 2 \cdot 200 = 400 .$$

Wir müssen also damit rechnen, daß ein gegebener Anfangsfehler in A und b sich um einen Faktor 400 verstärkt in x auswirkt.

Wir betrachten das Beispiel nun geometrisch. Die Lösung des Gleichungssystems läßt sich auch graphisch als Schnittpunkt von zwei Geraden bestimmen.



Grafische Darstellung des Modellproblems
Durch kleine Verschiebung von L_2 geht x von $(0,1)$ nach $(1,0)$.

Eine kleine Änderung von A oder b bewirkt eine große Änderung der Koordinaten des Schnittpunktes.

Wir wollen die Fehlerbetrachtungen auf die Rundungsfehler anwenden. Bei Lösung von $Ax = b$ entstehen zunächst bei der Eingabe auf dem Rechner Rundungsfehler $\Delta A = A - \tilde{A}$, $\Delta b = b - \tilde{b}$. Wir führen die Maschinengenauigkeit

$$\text{eps} = \max_{x \neq 0} \left| \frac{x - \tilde{x}}{x} \right| \tag{6.1}$$

ein, wo x alle reellen Zahlen durchläuft und $\tilde{x}$ die Rundung von x bedeutet. Bei m-stelliger dezimaler Gleitpunktarithmetik ist eps $\sim 10^{-m}$; der genaue Wert hängt von Zahldarstellung und Rundungsoperation ab. Der Exponentenbereich unserer Maschine wird als unendlich angenommen.

Es ist dann

$$\frac{\|\Delta A\|}{\|A\|} \sim \text{eps} \quad , \quad \frac{\|\Delta b\|}{\|b\|} \sim \text{eps} \; .$$

Die Anwendung des Satzes verlangt nun zunächst einmal

$$k(A)\,\text{eps} \; < 1 \; . \tag{6.2}$$

Ist dies der Fall, so folgt aus dem Satz für die Lösung $\tilde{x}$ von $\tilde{A}\tilde{x} = \tilde{b}$ mit $\Delta x = x - \tilde{x}$

$$\frac{\|\Delta x\|}{\|x\|} \sim k(A) \;\; \text{eps} \; . \tag{6.3}$$

Wir nennen dies den unvermeidbaren Fehler. Er ist nicht durch den verwendeten Algorithmus zur Lösung von $Ax = b$ bedingt, sondern allein durch die Maschinengenauigkeit und die Kondition von A bestimmt.

Wir untersuchen nun den Rundungsfehlereinfluß bei der Lösung über- und unterbestimmter Systeme.

Die Bildung von Matrizen wie A^*A, AA^* ist nicht unproblematisch.

1) Wir betrachten das Beispiel

$$A = \begin{pmatrix} 1 & 1 \\ \varepsilon & 0 \\ 0 & \varepsilon \end{pmatrix} \quad , \quad A^*A = \begin{pmatrix} 1+\varepsilon^2 & 1 \\ 1 & 1+\varepsilon^2 \end{pmatrix} \; .$$

Ist ε so klein, daß $\varepsilon^2 <$ eps, so kann A^*A auf der Maschine nicht zuverlässig berechnet werden.

2) Sei $n = m$ und A invertierbar. Mit $\|A\| = (\rho(A^*A))^{1/2}$ erhalten wir dann

$$\begin{aligned} &\|A^*A\| = \rho((A^*A)^2)^{1/2} = \rho(A^*A) = \|A\|^2 \; , \\ &\|(A^*A)^{-1}\| = \|A^{-1}\|^2 \end{aligned}$$

und damit

$$k(A^*A) = k(A)^2 \; .$$

Ist also $k(A) \gg 1$, so ist $k(A^*A) \gg k(A)$. Die Kondition von A^*A ist dann viel schlechter als die von A.

Im Gegensatz zu den Normalgleichungen haben (5.1), (5.2) vernünftige Kondition. Betrachten wir wieder den Fall $n = m$. Für $A = QR$ ist

$$\|A\| = \|QR\| = \max_{x \neq 0} \frac{\|QRx\|}{\|x\|} = \max_{x \neq 0} \frac{\|Rx\|}{\|x\|} = \|R\|$$

und entsprechend für A^{-1}. Also $k(A) = k(R)$, d.h. die Quadrierung der Kondition findet nicht statt.

Kapitel 2

Lineare Optimierung

2.1 Ein einführendes Beispiel

Wir wenden uns nun einem praxisorientierten Gebiet der Numerischen Mathematik zu. In der Linearen Optimierung werden lineare Zielfunktionen unter linearen Bedingungen optimiert. Häufig ist dies die Gewinnmaximierung unter gegebenen Einschränkungen durch Produktionskapazität, Lagervorrat und gesetzliche Bestimmungen. Diese Anwendung macht die lineare Optimierung zu einem der wichtigsten mathematischen Methoden in der Betriebswirtschaft. Wir werden sie zunächst informell behandeln. Wir diskutieren das folgende Beispiel:

Ein Chemiekonzern stellt die Chemikalien C_1 und C_2 her. Bei der Produktion fallen die Schadstoffe S_1, S_2 und S_3 an (laut Tabelle). Der maximale Schadstoffausstoß pro Tag ist begrenzt. Erarbeiten Sie einen Produktionsplan, der einen maximalen Gewinn sicherstellt.

Tabelle 1:

	$C_1(^{mg}/\ell)$	$C_2(^{mg}/\ell)$	Höchstgrenze $(^{mg}/d)$
S_1	20	10	8000
S_2	4	5	2000
S_3	6	15	4500
Gewinn (DM/ℓ)	16	32	

Wir formulieren die Optimierungsaufgabe wie folgt: Seien c_1, c_2 die pro Tag von C_1, C_2 produzierten Mengen in Liter. Es muß gelten

$$\begin{array}{rcrcrcl} c_1 & \geq & 0 & & c_1 & \geq & 0 \\ c_2 & \geq & 0 & \text{oder} & c_2 & \geq & 0 \\ 20c_1 + 10c_2 & \leq & 8000 & & 8000 - 20c_1 - 10c_2 & \geq & 0 \\ 4c_1 + 5c_2 & \leq & 2000 & & 2000 - 4c_1 - 5c_2 & \geq & 0 \\ 6c_1 + 15c_2 & \leq & 4500 & & 4500 - 6c_1 - 15c_2 & \geq & 0 \end{array}$$

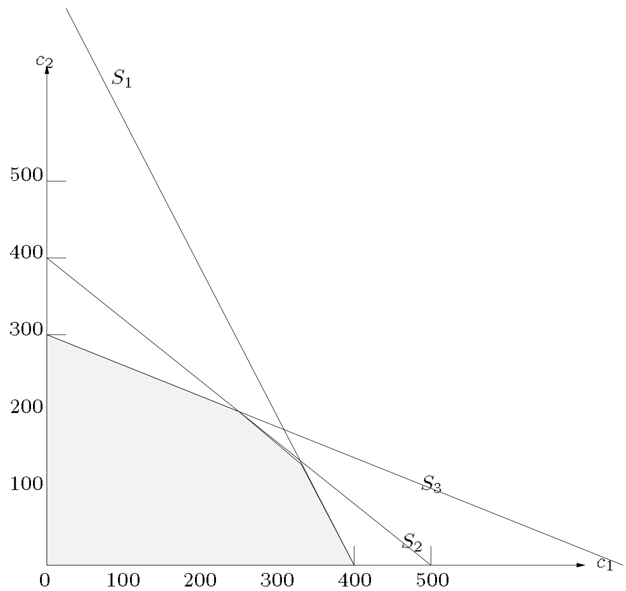
Mit den Definitionen

$$c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \quad , \quad b = \begin{pmatrix} 8000 \\ 2000 \\ 4500 \end{pmatrix} \quad , \quad A = \begin{pmatrix} -20 & -10 \\ -4 & -5 \\ -6 & -15 \end{pmatrix}$$

muß daher gelten

$$c \geq 0 \qquad \text{und} \qquad b + Ac \geq 0 \, .$$

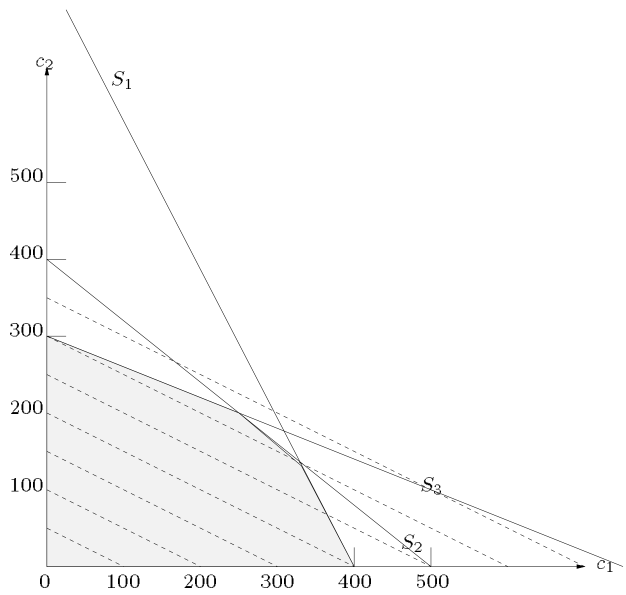
Ungleichungen zwischen Vektoren sind immer komponentenweise gemeint. Wir zeichnen den durch diese Ungleichungen bestimmten Lösungsbereich M.



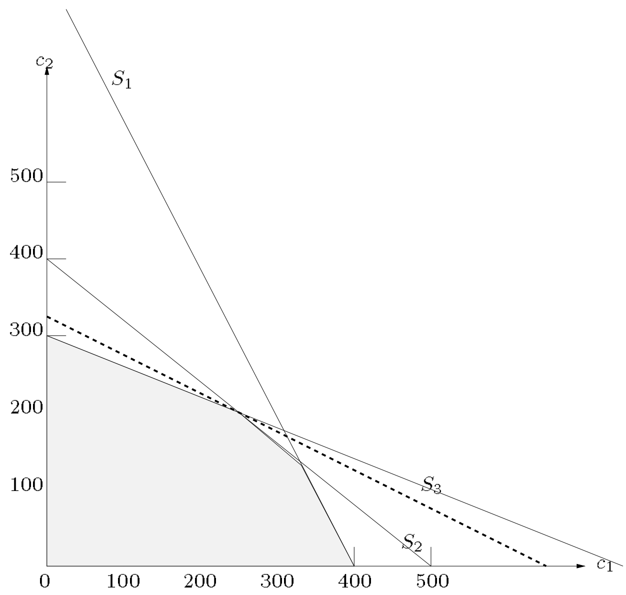
Es gilt: Gewinn (in DM/d) = $16c_1 + 32c_2$. Es ist also genau dann möglich, einen Gewinn von g DM zu erzielen, wenn die durch $16c_1 + 32c_2 = g$ bestimmte Gerade $G(g)$ einen gemeinsamen Punkt mit M besitzt.

Wir suchen also den größten Wert von g, so daß $G(g)$ einen Schnittpunkt mit L hat.

Im folgenden Bild ist $G(g)$ für einige Werte von g gezeichnet worden.



Für steigendes g verlagert sich die Linie nach oben. Die höchste Linie, die noch einen Schnittpunkt mit M hat, ist im folgenden gezeichnet:



Wir bestimmen den Schnittpunkt von S_2, S_3 als Lösung des Gleichungssystems

$$\begin{aligned} 4c_1 + 5c_2 &= 2000 \\ 6c_1 + 15c_2 &= 4500 \end{aligned}$$

als $c = (250, 200)$.

Es ist anschaulich klar, daß es immer eine Ecke von M gibt, in der ein optimales Ergebnis erzielt wird. "Ecken" sind dabei Lösungen eines linearen Gleichungssystems in n Variablen, wenn eine Zielfunktion im $\mathbb{R}^n$ optimiert werden soll.

Wir können daher folgenden einfachen Algorithmus zur Lösung der linearen Optimierungsaufgabe angeben.

Für alle Schnittpunkte von n Ungleichungen:

> Stelle fest, ob der Schnittpunkt zulässig ist (d.h., ob alle Ungleichungen erfüllt sind).
>
> Bestimme unter allen zulässigen Schnittpunkten das Zielfunktionsmaximum.

Zur Bestimmung der Schnittpunkte müssen bei m Ungleichungen $\begin{pmatrix} m \\ n \end{pmatrix}$

Gleichungssysteme der Ordnung n gelöst werden. Da typischerweise n und m groß sind (einige hundert), ist dieser Algorithmus aus Komplexitätsgründen nicht durchführbar.

Bemerkung zu der Anzahl $\begin{pmatrix} m \\ n \end{pmatrix}$: Hier bedeutet n die Anzahl der Variablen, m die Anzahl <u>aller</u> Ungleichungen, also einschließlich der Vorzeichenrestriktionen. In unserem Beispiel ist also $n = 2$, $m = 5$, also $\begin{pmatrix} m \\ n \end{pmatrix} = 5!/(2!3!) = 10$. In der Tat finden sich in der Abbildung auf S. 38 10 Schnittpunkte der 5 den Bereich M definierenden Geraden.

2.2 Die allgemeine lineare Optimierungsaufgabe

Wir wollen nun die allgemeine lineare Optimierungsaufgabe formulieren.

Sei A eine reelle (m,n)-Matrix und $c \in \mathbb{R}^n$. Sei

$$M = \{x \in \mathbb{R}^n \; : \; x_j \geq 0 \; , \; j = 1, \ldots, n_0 \; , \; \sum_{j=1}^{n} a_{i,j} x_j \leq b_i \; , \; i = 1, \ldots, m_0 \; , \tag{2.1}$$

$$\left. \sum_{j=1}^{n} a_{i,j} x_j = b_i \; , \; i = m_0 + 1, \ldots, m \right\} .$$

Die allgemeine lineare Optimierungsaufgabe lautet nun: Minimiere

$$z(x) = \sum_{j=1}^{n} c_j x_j \tag{2.2}$$

in M! Man nennt M die Menge der zulässigen Punkte und jedes $x \in M$ zulässig. Die lineare Funktion (2.2) heißt Zielfunktion.

Die Aufgabe (2.1), (2.2) heißt in Normalform, wenn $n_0 = n$ und $m_0 = 0$. Dann kann man M in der Form

$$M = \{x \in \mathbb{R}^n : x \geq 0 \; , \quad Ax = b\} \tag{2.3}$$

mit einer (m,n)-Matrix A und $m \leq n$ schreiben. Ungleichungen zwischen Vektoren werden hier und unten immer komponentenweise verstanden. Man kann (2.1) immer in der Form (2.3) schreiben. Ist $n_0 < n$, so setzt man

$$x_j = y_j - z_j \quad , \quad j = n_0 + 1, \ldots, n \quad , \quad y_j \; , \; z_j \geq 0 \; .$$

Ist $m_0 > 0$, treten also Ungleichungen auf, so führt man die "Schlupfvariablen"

$$u_i = b_i - \sum_{j=1}^{n} a_{ij} x_j \quad , \quad i = 1, \ldots, m_0$$

ein. Dann hat man ein Problem in Normalform mit den nicht negativen Variablen

$$\begin{array}{ll} x_j & , \; j = 1, \ldots, n_0 \\ y_j, z_j & , \; j = n_0 + 1, \ldots, n \\ u_i & , \; i = 1, \ldots, m_0 \end{array}$$

und den Gleichungen

$$\begin{aligned} u_i + \sum_{j=1}^{n_0} a_{i,j}x_j \quad &+ \quad \sum_{j=n_0+1}^{n} a_{i,j}(y_j - z_j) = b_i \ , \quad i = 1, \ldots, m_0 \\ \sum_{j=1}^{n_0} a_{i,j}x_j \quad &+ \quad \sum_{j=n_0+1}^{n} a_{i,j}(y_j - z_j) = b_i \ , \quad i = m_0 + 1, \ldots, m \ . \end{aligned}$$

Nun führen wir einige geometrische Begriffe ein.

1) Eine Menge $U \subseteq \mathbb{R}^n$ heißt konvex, wenn mit $x, y \in U$ auch $\lambda x + (1-\lambda)y \in U$ für $0 \leq x \leq 1$, d.h. wenn mit je zwei Punkten deren Verbindungsgerade in U liegt. Z.B. sind die Mengen M aus (2.1), (2.3) konvex.

2) Sei $U \subseteq \mathbb{R}^n$ konvex. $x \in U$ heißt Ecke von U, wenn x nicht im Innern einer Verbindungsgeraden zweier Punkte von U liegt. Mit anderen Worten: Es gilt nicht $x = \lambda y + (1-\lambda)z$ mit $y, z \in U,\ y \neq z,\ 0 < \lambda < 1$.

3) Sind $u_i \in \mathbb{R}^n$ und $\lambda_i \geq 0,\ \sum\limits_i \lambda_i = 1$, so heißt $\sum\limits_i \lambda_i u_i$ Konvexkombination der u_i. Die Menge aller Konvexkombinationen heißt konvexe Hülle der u_i.

Von dem einleitenden Beispiel in §1 ist klar, daß die Ecken von M eine besondere Rolle spielen. Wir wollen diese Ecken algebraisch charakterisieren. Im folgenden nehmen wir immer an, daß A den Rang m hat, und für M nehmen wir immer die Normalform (2.3).

Definition 2.2.1 *Sei A eine reelle (m,n)-Matrix. Eine Lösung x von $Ax = b$ heißt Basis-Lösung, wenn es eine Teilmenge I von $\{1, \ldots, n\}$ mit genau m Elementen gibt, so daß gilt:*

i) $\quad x_i = 0 \ , \quad i \notin I$

ii) $\quad a_i \ , \ i \in I$ *sind linear unabhängig.*

I heißt Basis von x.

Satz 2.2.1 *$x \in \mathbb{R}^n$ ist genau dann Ecke von M, wenn x zulässige Basis-Lösung von $Ax = b$ ist.*

Beweis:

a) Sei x Ecke von M. Sei I die Menge der Indizes i mit $x_i > 0$. Wären a_i, $i \in I$ linear abhängig, so gäbe es Zahlen α_i, $i \in I$ mit

$$\sum_{i \in I} \alpha_i a_i = 0 \ , \quad \sum_{i \in I} |\alpha_i| > 0 \ .$$

Wir könnten die Punkte $x^\pm$ mit den Komponenten

$$\begin{array}{rcll} x_i^\pm & = & x_i \pm \varepsilon \alpha_i & , \quad i \in I \ , \\ x_i^\pm & = & 0 & , \quad i \notin I \end{array}$$

bilden. Es wäre $Ax^\pm = b$ und $x^\pm \geq 0$ für hinreichend kleines ε, also $x^\pm \in M$. Außerdem wäre $x = \frac{1}{2}(x^+ + x^-)$. Dies ist nicht möglich, da x Ecke ist. Also sind die a_i für $i \in I$ linear unabhängig. Insbesondere hat I höchstens m Elemente. Hat I weniger als m Elemente, so können wir I zu einer m-elementigen Menge ergänzen. x ist damit zulässige Basis-Lösung mit Basis I.

b) Sei x umgekehrt zulässige Basis-Lösung mit Basis I. Wäre x nicht Ecke, so gäbe es x^1, $x^2 \in M$, $x^1 \neq x^2$, und λ mit $0 < \lambda < 1$, so daß $x = \lambda x^1 + (1 - \lambda) x^2$. Für $i \notin I$ müßte dann $x_i^1 = x_i^2 = 0$ sein, und wegen

$$\sum_{i \in I} x_i^1 a_i = \sum_{i \in I} x_i^2 a_i$$

müßte $x_i^1 = x_i^2$ auch für $i \in I$ sein. Im Widerspruch zur Annahme wäre also $x^1 = x^2$.

$\square$

Satz 2.2.2 *Sei M beschränkt. Dann ist M die konvexe Hülle seiner Ecken.*

Beweis: Es ist zu zeigen, daß jedes $x \in M$ Konvexkombination der Ecken von M ist.

Sei also $x \in M$ und p die Anzahl der positiven Komponenten von x.

Ist x Ecke, so sind wir fertig. Andernfalls sind die p Vektoren a_i, $x_i > 0$ linear abhängig, d.h. es gibt $w \in \mathbb{R}^n$, $w \neq 0$ mit $Aw = 0$ und $w_i = 0$ für $x_i = 0$. Da M beschränkt ist, muß w sowohl positive als auch negative Komponenten haben. Also kann man t_1, $t_2 > 0$ so finden, daß

$$x^1 = x - t_1 w \quad , \quad x^2 = x + t_2 w$$

jeweils höchstens $p-1$ positive Komponenten haben und x^1, $x^2 \in M$. x ist Konvexkombination von x^1, x^2, denn es ist

$$x = \frac{t_2}{t_1 + t_2} x^1 + \frac{t_1}{t_1 + t_2} x^2 \ .$$

x^1 und x^2 haben höchstens $p-1$ positive Komponenten. Wir haben damit das Problem auf Punkte mit $p-1$ positiven Komponenten reduziert. Durch Wiederholung dieses Prozesses landet man entweder bei Ecken oder bei $p = 0$; der letztere Fall bedeutet aber ebenfalls eine Ecke, weil eine leere Menge von Vektoren linear unabhängig ist.

□

Satz 2.2.3 *M sei beschränkt und nicht leer. Dann ist die lineare Optimierungsaufgabe lösbar, und es gibt sogar eine Ecke von M, die Lösung ist.*

Beweis: Die Zielfunktion z nimmt als stetige Funktion auf dem Kompaktum M ihr Minimum an, etwa in $x^0 \in M$. Seien $x^1, \ldots, x^k$ die Ecken von M. Nach Satz 2.2.2 ist x^0 Konvexkombination von $x^1, \ldots, x^k$, also

$$x^0 = \sum_{i=1}^{k} \lambda_i x^i \quad , \quad \lambda_i \geq 0 \quad , \quad \sum_{i=1}^{k} \lambda_i = 1 \ .$$

Wäre nun $z(x^0) < z(x^i)$ für alle i, so wäre

$$z(x^0) = \sum_{i=1}^{k} \lambda_i z(x^i) > \sum_{i=1}^{k} \lambda_i z(x^0) = z(x^0) \ ,$$

und dies ist ein Widerspruch. Also gibt es ein i mit $z(x^0) = z(x^i)$, und die Ecke x^i ist Lösung.

$\square$

2.3 Das Simplex-Verfahren: Phase II

Das Simplex-Verfahren ist das Standard-Verfahren zur numerischen Lösung linearer Optimierungsaufgaben. Es wurde gegen Ende des zweiten Weltkriegs von Dantzig entwickelt und zur optimalen Ladungsverteilung von der amerikanischen Marine benutzt. Bei diesen Anwendungen war die Menge M ein Simplex. Wir beschreiben das sogenannte revidierte Simplex-Verfahren, welches für sehr große Probleme besonders geeignet ist.

Wir beschreiben das Verfahren für die Normalform

$$\text{Minimiere} \quad z(x) = \sum_{i=1}^{n} c_i x_i \;\text{ in }\; M = \{x \in \mathbb{R}^n : x \geq 0 \;\;, Ax = b\} \tag{3.1}$$

Wie immer soll die (m,n)-Matrix A den Rang m besitzen. Ecken x von M sind dann dadurch gekennzeichnet, daß die Vektoren a_i mit $x_i > 0$ linear unabhängig sind. Natürlich kann es nicht mehr als m solcher Komponenten x_i geben. Sind es genau m, so heißt die Ecke nicht entartet, andernfalls entartet. Für nicht entartete Ecken gibt es immer genau eine Basis $I = \{i : x_i > 0\}$. Diese enthält genau m Elemente, und die Basis-Vektoren a_i, $i \in I$ sind linear unabhängig. Wir wollen annehmen, daß keine Ecke von M entartet ist.

Das Simplex-Verfahren konstruiert eine endliche Folge von Ecken von M mit abnehmenden Werten von z und endet mit einer Lösung oder der Information, daß das Problem nicht lösbar ist. Jede Ecke kann als Anfang dieser Folge gewählt werden. Die Berechnung einer Ausgangsecke von M ist Gegenstand der Phase I des Simplex-Verfahrens. Bei der jetzt zu schildernden Phase II nehmen wir an, wir hätten eine Ecke x^0 bereits gefunden. Es sei I die Basis von x^0 und J die Menge der nicht in I liegenden Indizes oder die Nicht-Basis-Indizes. Es ist also $x_i^0 > 0$ für $i \in I$, $x_j^0 = 0$ für $j \in J$, und a_i, $i \in I$, sind linear unabhängig. Wir können $Ax = b$ nach den Basis-Variablen x_i, $i \in I$ auflösen und erhalten

$$z = z(x^0) + \sum_{j \in J} p_j x_j \tag{3.2}$$

$$x_i = x_i^0 + \sum_{j \in J} c_{i,j} x_j \quad , \quad i \in I \ . \tag{3.3}$$

Ist $A_I = (a_i)_{i \in I}$, $A_J = (a_j)_{j \in J}$, so gilt für die $(m, n-m)$-Matrix $C = (c_{ij})_{i \in I, j \in J}$

$$C = -A_I^{-1} A_J \ . \tag{3.4}$$

Die Koeffizienten von (3.2)-(3.3) fassen wir ins Simplex-Tableau der Ecke x^0 zusammen:

$$\begin{array}{c|c|c|}
 & & J \\
\hline
 & z(x^0) & p_j \\
\hline
I & x_i^0 & c_{i,j} \\
\hline
\end{array} \tag{3.5}$$

Nun kommt die Beschreibung des Simplex-Schrittes, welcher die Ecke x^0 in eine neue Ecke x^1 mit kleinerem Wert der Zielfunktion überführt. Er besteht aus folgenden Teilschritten:

1. Suche $j_0 \in J$ mit $p_{j_0} < 0$.

Gibt es kein solches, so ist $p_j \geq 0$ für alle $j \in J$ und damit nach (3.2) $z(x) \geq z(x_0)$ für alle $x \in M$. Also ist x^0 bereits Lösung und das Verfahren beendet.

2. Suche nach einem $i_0 \in I$ mit $c_{i_0,j_0} < 0$ und

$$\frac{x_{i_0}^0}{-c_{i_0,j_0}} \leq \frac{x_i^0}{-c_{i,j_0}} \quad \text{für alle} \quad i \in I \ \text{ mit } \ c_{i,j_0} < 0 \ .$$

Gibt es kein solches i_0, so ist $c_{i,j_0} \geq 0$ für alle $i \in I$, und der Punkt $x(t) = x^0 + te_{j_0}$ ist wegen (3.3) zulässig für alle $t \geq 0$, während wegen (3.2) $z(x(t)) \to -\infty$ für $t \to +\infty$. Also ist z nach unten nicht beschränkt auf M, die Optimierungsaufgabe unlösbar, und das Verfahren beendet.

3. Konstruktion der neuen Ecke x^1.

Gibt es $i \in I$ mit $c_{i,j_0} < 0$, so ist wegen (3.3) $x(t) = x^0 + te_{j_0} \in M$ genau für $x_i(t) = x_i^0 + tc_{i,j_0} \geq 0$ für alle diese i, d.h. für

$$t \leq \frac{x_i^0}{-c_{i,j_0}} \quad \text{für } \ i \in I \ \text{ mit } \ c_{i,j_0} < 0 \; .$$

Nach Wahl von i_0 ist $t_0 = x_{i_0}^0/(-c_{i_0,j_0})$ das größte t mit dieser Eigenschaft. Wir setzen

$$x^1 = x(t_0) = x^0 + t_0 e_{j_0} \; .$$

Es ist

$$\begin{array}{rcl} z(x^1) & = & z(x^0) + t_0 p_{j_0} < z(x^0) \; , \\ x_{i_0}^1 & = & 0 \quad , \quad x_{j_0}^1 = t_0 > 0 \; . \end{array}$$

Wir zeigen, daß x^1 eine Ecke mit Basis $I \setminus \{i_0\} \cup \{j_0\}$ ist, d.h. daß die Vektoren $a_i, \ i \in I \setminus \{i_0\} \cup \{j_0\}$ linear unabhängig sind. Wäre dies nicht der Fall, so gäbe es Zahlen $\alpha_i, \ i \in I \setminus \{i_0\} \cup \{j_0\}$, die nicht alle verschwinden, so daß

$$\sum_{i \in I \setminus \{i_0\}} \alpha_i a_i + \alpha_{j_0} a_{j_0} = 0 \; .$$

Nach (3.4) ist $A_I C = A_J$. Die j_0-te Spalte hiervon lautet

$$\sum_{i \in I} c_{i,j_0} a_i = a_{j_0} \; .$$

Damit haben wir zwei Darstellungen von a_{j_0} durch die linear unabhängigen Vektoren $a_i, \ i \in I$. In der ersten verschwindet der Koeffizient von a_{i_0}, in der zweiten ist er $c_{i_0,j_0} \neq 0$. Dieser Widerspruch zeigt, daß die a_i mit $i \in I \setminus \{i_0\} \cup \{j_0\}$ linear unabhängig sind und damit x^1 eine Ecke mit der Basis $I \setminus \{i_0\} \cup \{j_0\}$ ist.

4. Berechnung des neuen Tableaus.

Dazu löst man $Ax = b$ oder (3.3) nach den neuen Basis-Variablen $x_i, \ i \in I \setminus \{i_0\} \cup \{j_0\}$ auf. Aus der Gleichung (3.3) für $i = i_0$ erhalten wir zunächst

$$x_{j_0} = (x_{i_0} - x_{i_0}^0 - \sum_{j \in J \setminus \{j_0\}} c_{i_0,j} x_j)/c_{i_0,j_0}$$

und dann aus den restlichen Gleichungen von (3.3)

$$
\begin{aligned}
x_i &= x_{i_0}^0 + \sum_{j \in J \setminus \{j_0\}} c_{i,j} x_j + c_{i,j_0} x_{j_0} \\
&= x_{i_0}^0 + \sum_{j \in J \setminus \{j_0\}} c_{i,j} x_j + \frac{c_{i,j_0}}{c_{i_0,j_0}} \left(x_{i_0} - x_{i_0}^0 - \sum_{j \in J \setminus \{j_0\}} c_{i_0,j} x_j \right) \\
&= x_{i_0}^0 - \frac{c_{i,j_0}}{c_{i_0,j_0}} x_{i_0}^0 + \frac{c_{i,j_0}}{c_{i_0,j_0}} x_{i_0} + \sum_{j \in J \setminus \{j_0\}} \left(c_{i,j} - \frac{c_{i,j_0}}{c_{i_0,j_0}} c_{i_0,j} \right) x_j
\end{aligned}
$$

für $i \in I \setminus \{i_0\}$. Auch die Zielfunktion muß in den neuen Nicht-Basis-Variablen ausgedrückt werden. Wir erhalten

$$
\begin{aligned}
z &= z(x^0) + \sum_{j \in J \setminus \{j_0\}} p_j x_j + p_{j_0} x_{j_0} \\
&= z(x^0) - \frac{p_{j_0}}{c_{i_0,j_0}} x_{i_0}^0 + \frac{p_{j_0}}{c_{i_0,j_0}} x_{i_0} + \sum_{j \in J \setminus \{j_0\}} \left(p_j - \frac{p_{j_0}}{c_{i_0,j_0}} c_{i_0,j} \right) x_j \ .
\end{aligned}
$$

Das (i_0, j_0)-Element im Simplex-Tableau heißt Pivotelement, die zugehörige Zeile bzw. Spalte Pivotzeile bzw. Pivotspalte. Das Simplex-Tableau für x^1 hat die Form

		$j \in J \setminus \{j_0\}$	i_0
	$z(x^0) - \frac{p_{j_0}}{c_{i_0,j_0}} x_{i_0}^0$	$p_j - \frac{p_{j_0}}{c_{i_0,j_0}} c_{i_0,j}$	$\frac{p_{j_0}}{c_{i_0,j_0}}$
$i \in I \setminus \{i_0\}$	$x_i^0 - \frac{c_{i,j_0}}{c_{i_0,j_0}} x_{i_0}^0$	$c_{i,j} - \frac{c_{i,j_0}}{c_{i_0,j_0}} c_{i_0,j}$	$\frac{c_{i,j_0}}{c_{i_0,j_0}}$
j_0	$-\frac{x_{i_0}^0}{c_{i_0,j_0}}$	$-\frac{c_{i_0,j}}{c_{i_0,j_0}}$	$\frac{1}{c_{i_0,j_0}}$

Dabei schreibt man die neue Zeile j_0 (Spalte i_0) zweckmäßigerweise in die alte Zeile i_0 (Spalte j_0).

Sind, wie vorausgesetzt, alle Ecken von M nicht entartet, so ist $t_0 > 0$, und die Zielfunktion nimmt bei jedem Schritt echt ab. Das Simplex-Verfahren bricht also nach endlich vielen Schritten ab, entweder mit der Lösung, oder mit der Aussage, daß z auf M nicht nach unten beschränkt ist.

Wir schreiben nun ein Programm, das den Übergang von einem Tableau zum nächsten bewerkstelligt. Wir werden sehen, daß das ganze Tableau, also auch erste Spalte und Zeile, einheitlich behandelt werden können. Deshalb nehmen wir das Tableau in der Form

$$\begin{array}{c|ccc} c_{0,0} & c_{0,1} & \cdots & c_{0,q} \\ \hline c_{1,0} & c_{1,1} & \cdots & c_{1,q} \\ \vdots & \vdots & & \vdots \\ c_{m,0} & c_{m,1} & \cdots & c_{m,q} \end{array}$$

an mit $q = n - m$. Die Zeilenindizes $1, \ldots, m$ entsprechen also den Basis-Indizes $i_1, \ldots, i_n$, die Spaltenindizes $1, \ldots, q$ den Nicht-Basis-Indizes $j_1, \ldots, j_q$.

austausch (C, i_0, j_0, m, q)

/ ⋆ Berechnet nach Bestimmung von i_0, j_0 das neue Tableau.
Schreibt den neuen Basis-Index j_{j_0} in Zeile i_0 und den
neuen Nicht-Basis-Index i_{i_0} in die Spalte j_0 / ⋆

{ $c_{i_0,j_0} = 1/c_{i_0,j_0}$;
for $i = 0, \ldots, m$ { if $(i \neq i_0) c_{i,j_0} = c_{i,j_0} * c_{i_0,j_0}$;}
for $i = 0, \ldots, m$
for $j = 0, \ldots, q$ { if $(i \neq i_0$ und $j \neq j_0)$ $c_{i,j} = c_{i,j} - c_{i,j_0} * c_{i_0,j}$;}
for $j = 0, \ldots, q$ { if $(j \neq j_0)$ $c_{i_0,j} = -c_{i_0,j} * c_{i_0,j_0}$;}
}

Der Austauschschritt benötigt $qm+$ $\mathrm{O}(m+q)$ flops.

Mit Hilfe der Austauschroutine kann man nun leicht ein Programm für die Phase II des Simplex-Verfahrens schreiben:

simplex (C, I, J, m, q)

$/\star$ Bei Aufruf enthält C das Tableau der Ausgangs-Ecke, I die Basis-, J die Nicht-Basis-Indizes. Nach Ablauf steht das Minimum der Zielfunktion in $c_{0,0}$, die Lösung in $c_{i,0}$, $i = 1, \ldots, m$ und die Basis-Indizes der Lösung in I $/\star$

```
{  while      (∃ j ∈ {1,...,q} : c_{0,j} < 0)
           {  Wähle j_0;
              Wähle i_0 ∈ {1,...,m}, so daß c_{i,0}/(−c_{i,j_0}) minimal für
              i = i_0 unter allen i mit c_{i,j_0} < 0;
              Falls es kein solches i_0 gibt,
              breche ab mit Meldung "Problem nicht lösbar"
              I = (I \ {i_{i_0}}) ∪ {j_{j_0}}; J = (J \ {j_{j_0}}) ∪ {i_{i_0}};
              austausch (C, i_0, j_0, m, q);
           }
}
```

Der Austauschschritt kann auch zum Invertieren von Matrizen benutzt werden. Sei C eine $(n+1, n+1)$-Matrix und $y = Cx$, also

$$\begin{array}{lcllc} y_0 & = & c_{0,0}x_0 & + \cdots + & c_{0,n}x_n \\ \vdots & & & & \\ y_n & = & c_{n,0}x_0 & + \cdots + & c_{n,n}x_n \ . \end{array}$$

Wir lösen die nullte Gleichung nach x_0 auf und setzen x_0 in die restlichen Gleichungen ein. Es entsteht

$$\begin{array}{lcllllll} x_0 & = & c'_{0,0}y_0 & + & c'_{0,1}x_1 & + \cdots + & c'_{0,n}x_n \\ y_1 & = & c'_{1,0}y_0 & + & c'_{1,1}x_1 & + \cdots + & c'_{1,n}x_n \\ \vdots & & & & & & \\ y_n & = & c'_{n,0}y_0 & + & c'_{n,1}x_1 & + \cdots + & c'_{n,n}x_n \end{array}$$

mit neuen Elementen $c'_{i,j}$, die wir durch den Aufruf

$$\text{austausch } (C, 0, 0, n, n)$$

berechnen können (falls $c_{0,0} \neq 0$). Danach lösen wir die erste Gleichung nach x_1 auf und setzen x_1 in die restlichen Gleichungen ein. Dann sind x_0, x_1,

$y_2, \ldots, y_n$ durch y_0, y_1, $x_2, \ldots, x_n$ ausgedrückt. So fortfahrend können wir schließlich $x_0, \ldots, x_n$ durch $y_0, \ldots, y_n$ ausdrücken. Die Koeffizienten dieser Formeln sind dann gerade die Elemente von C^{-1}. Das Programm

for $i = 0, \ldots, n$
austausch (C, i, i, n, n)

invertiert also C. Das geht natürlich nur gut, wenn alle Pivotelemente $\neq 0$ sind.

2.4 Das Simplex-Verfahren: Phase I

Die in §3 beschriebene Phase II des Simplex-Verfahrens setzt voraus, daß wir eine Ecke x^0 von M kennen. Eine solche Ecke liefert uns die Phase I. Dabei nehmen wir wieder an, daß M in der Normalform

$$M = \{x \in \mathbb{R}^n : x \geq 0 \ , \quad Ax = b\}$$

vorliegt mit einer (m, n)-Matrix A. Wir können zusätzlich $b \geq 0$ annehmen.

Wir führen nun ein Hilfsproblem in den $n+m$ Variablen $x_1, \ldots, x_n, y_1, \ldots, y_m$ ein. Seine zulässige Menge ist

$$\tilde{M} = \{x, y : x \in \mathbb{R}^n, \ y \in \mathbb{R}^m, \ x, y \geq 0, \ Ax + y = b\} \ ,$$

und seine Zielfunktion

$$\tilde{z}(x, y) = \sum_{i=1}^{m} y_i \ .$$

Wir zeigen nun: M ist nicht leer genau dann, wenn das Hilfsproblem eine Lösung mit $y = 0$ besitzt. Ist nämlich $x \in M$, so ist $\begin{pmatrix} x \\ 0 \end{pmatrix}$ Lösung des Hilfsproblems. Ist umgekehrt $\begin{pmatrix} x \\ 0 \end{pmatrix}$ Lösung des Hilfsproblems, so ist $x \in M$.

Das Hilfsproblem ist - für $b > 0$ - genau von der Form, die wir für die Phase II vorausgesetzt haben. Eine Ecke von $\tilde{M}$ ist $\begin{pmatrix} 0 \\ b \end{pmatrix}$, die zugehörigen Basis-Variablen sind $y = b - Ax$, und die Zielfunktion ist

$$\tilde{z}(x, y) = e^*(b - Ax) \ , \qquad e^* = (1, \ldots, 1) \ .$$

Man kann nun die Phase II des Simplex-Verfahrens für das Hilfsproblem durchführen. Endet dies mit einer Lösung mit $y = 0$, so hat man eine Startecke von M gefunden. Andernfalls ist M leer.

Kapitel 3

Nichtlineare Gleichungen

3.1 Existenz von Lösungen

Sei $f : K^n \to K^n$ eine Abbildung. Gesucht ist ein $\overline{x} \in K^n$ mit $f(\overline{x}) = 0$.

Beispiele:

1) $f(x) = x^2 - 2px - q$ für $K = \mathbb{C}$ oder $K = \mathbb{R}$. Sei $d = p^2 + q$. Im komplexen Fall gibt es zwei Lösungen für $d \neq 0$, eine Lösung für $d = 0$. Im reellen Fall gibt es für $d > 0$ zwei Lösungen, für $d = 0$ eine, für $d < 0$ überhaupt keine. Die Berechnung der Lösung $\overline{x}$ kann durch die Formel

$$x_{1,2} = p \pm \sqrt{d}$$

erfolgen. Die Auswertung dieser Formel ist aber im Hinblick auf Rundungsfehler keineswegs harmlos. Ist etwa $p > 0$, $|q| \ll p$, so ist $\sqrt{d} \sim p$, und bei der Berechnung von x_2 tritt Auslöschung auf. In diesem Fall ist es besser, x_2 nach der Formel $x_2 = -q/x_1$ zu berechnen.

2) $f(x) = x^3 + 3px - 2q$ für $K = \mathbb{C}$. Die Berechnung der - maximal drei - Lösungen kann durch die Cordani'schen Formeln erfolgen:

$$\begin{array}{rclcrclcrclc} x_1 & = & u + v & , & x_2 & = & \varepsilon_1 u + \varepsilon_2 v & , & x_3 & = & \varepsilon_2 u + \varepsilon_1 v & , \\ u & = & (q + \sqrt{d})^{1/3} & , & v & = & (q - \sqrt{d})^{1/3} & , & d & = & p^3 + q^2 & , \end{array}$$

$$\varepsilon_{1,2} = -\frac{1}{2}(1 \pm i\sqrt{3}) \, .$$

Auch hier tritt unter Umständen, nämlich für $|p| \ll |q|$, Auslöschung auf.

3) $f(x) = x - \tan x$, $K = \mathbb{R}$. Ein Blick auf den Graphen von $\tan x$ zeigt, daß es in jedem Intervall $((k - \frac{1}{2}), (k + \frac{1}{2})\pi)$, $k \in \mathbb{Z}$, genau eine Lösung gibt.

Ein primitives Verfahren zur Lösung von Gleichungen in $\mathbb{R}^1$ ist die Intervallhalbierung. Sei $f : \mathbb{R} \to \mathbb{R}$ stetig und $f(a)f(b) < 0$, $a < b$. Dann liegt in (a, b) sicher eine Nullstelle von f. Zu ihrer Berechnung verwenden wir den Algorithmus

$$
\begin{array}{lcl}
f_a = f(a) & ; & f_b = f(b); \\
\quad\text{while} & & (b - a \geq \varepsilon) \\
 & \{ & c = (b + a)/2; \\
 & & f_c = f(c); \\
 & & \text{if } (f_a f_c < 0) \;\; \{b = c; f_b = f_c;\} \\
 & & \text{else } \{a = c; f_a = f_c;\} \\
 & \} &
\end{array}
$$

Nach $n + 2$ Funktionsauswertungen hat er das Intervall (a, b), in dem eine Lösung liegt, um den Faktor 2^n verkürzt. Für einfache Funktionen f ist dies akzeptabel. Wir werden natürlich effizientere Methoden kennenlernen.

Das theoretische Hilfsmittel zur Lösung nichtlinearer Gleichungen sind Fixpunktsätze. $x \in \mathbb{R}^n$ heißt Fixpunkt einer Abbildung $g : \mathbb{R}^n \to \mathbb{R}^n$, falls $g(x) = x$. Jede Aufgabe $f(x) = 0$ läßt sich in der Form $g(x) = x$ schreiben. Man braucht ja nur $g(x) = f(x) + x$ zu setzen.

Satz 3.1.1 *(Fixpunktsatz von Brouwer): Sei $D \subseteq \mathbb{R}^n$ konvex und kompakt, $g : D \to D$ stetig. Dann besitzt g in D einen Fixpunkt.*

Beweis: Der Beweis für $n \geq 1$ findet sich in E. Burger, Einführung in der Theorie der Spiele, 2. Auflage, S. 162-165. Für $n = 1$ ist der Satz trivial: Sei $D = [a, b]$. Ist $g(a) = a$ oder $g(b) = b$, so besitzt g einen Fixpunkt. Andernfalls ist $g(a) > a$, $g(b) < b$. Die Funktion $f(x) = x - g(x)$ hat dann in $[a, b]$ einen Zeichenwechsel und damit eine Nullstelle, und diese ist Fixpunkt von g.

$\square$

Beispiele:

1) In $\mathbb{R}^2$ betrachten wir

$$g\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \sin(x_2 + e^{x_1}) \\ \cos(x_1 - e^{x_2}) \end{pmatrix}$$

Die Menge $D = \{x \in \mathbb{R}^2 : \|x\|_\infty \leq 1\}$ ist konvex und kompakt, und g ist dort stetig. Offenbar ist $g(D) \subseteq D$. Also hat g in D einen Fixpunkt.

2) Der Brouwer'sche Satz wird häufig auf Probleme der Volkswirtschaft angewendet. Ein Markt bestehe aus n Gütern mit Preisen $x_1, \ldots, x_n \geq 0$ und aus m Teilnehmern (Produzenten oder Konsumenten). Bei den Preisen $x = (x_1, \ldots, x_n)$ kauft oder produziert der ℓ-te Teilnehmer die Menge $f_i^\ell(x)$ von Gut i. Die totale Nachfrage bzw. Produktion von Gut i ist dann $f_i(x) = \sum_{\ell=1}^{m} f_i^\ell(x)$. Man sagt, der Markt sei im Gleichgewicht, wenn

$$f_i(x) \leq 0\ , \quad i = 1, \ldots, n, \quad f_i(x) = 0 \quad \text{falls} \ \ x_i > 0\ .$$

Wir machen folgende Voraussetzungen:

1) Die Nachfrage hängt nur von den relativen Preisen ab, d.h.

$$f_i^\ell(tx) = f_i^\ell(x) \quad , \quad t > 0\ .$$

2) Der Markt ist abgeschlossen, d.h.

$$\sum_{i=1}^{n} x_i f_i^\ell(x) = 0 \quad , \quad \ell = 1, \ldots, m$$

3) Die f_i^ℓ sind stetig.

Dann gibt es einen Preisvektor x, bei welchem der Markt im Gleichgewicht ist.

Zum Beweis setzen wir $D = \{x \in \mathbb{R}^n : x \geq 0,\ \sum_{i=1}^{n} x_i = 1\}$ und auf D

$$g_i(x) = \frac{x_i + \ \mathrm{Max}\ (0, f_i(x))}{1 + \sum_{k=1}^{n}\ \mathrm{Max}\ (0, f_k(x))}\ , \quad i = 1, \ldots, n\ .$$

Die Abbildung $g(x) = (g_1(x), \dots, g_n(x))$ ist dann stetig auf der konvexen kompakten Menge D und bildet D in sich ab. Nach dem Satz gibt es ein $x \in D$ mit $g(x) = x$, also

$$x_i = \frac{x_i + \text{ Max } (0, f_i(x))}{1 + \sum\limits_{k=1}^{n} \text{ Max } (0, f_k(x))} \quad , \quad i = 1, \dots, n \; .$$

Hieraus folgt, daß x Gleichgewichtspunkt ist (Aufgabe 26).

3.2 Iterationsverfahren

Iterationsverfahren gehören zu den wichtigsten Hilfsmitteln der Numerischen Mathematik. Sie berechnen mittels einer - oft sehr einfachen - Rekursionsformel eine Folge von Näherungen, welche gegen die gesuchte Lösung konvergiert. Grundlage vieler Iterationsverfahren ist der Fixpunktsatz für kontrahierende Abbildungen.

Definition 3.2.1 *Sei $D \subseteq K^n$ und $g : D \to K^n$ eine Abbildung. g heißt kontrahierend in D (bezüglich der Norm $\|\cdot\|$ in K^n), wenn es eine Konstante $q < 1$ gibt mit*

$$\|g(x) - g(y)\| \le q\|x - y\|$$

für alle x, $y \in D$. q heißt Lipschitzkonstante von g in D.

Satz 3.2.1 *Sei $D \subseteq \mathbb{R}^n$ konvex und $g : D \to \mathbb{R}^n$ differenzierbar. Sei*

$$q = \sup_{x\in D} \|g'(x)\| < 1 \ .$$

Dann ist g in D (bezüglich $\|\cdot\|$) kontrahierend. Dabei ist g' die Jacobi-Matrix zu g, also

$$g = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & ,\ldots, & \frac{\partial g_1}{\partial x_n} \\ \vdots & & \\ \frac{\partial g_n}{\partial x_1} & ,\ldots, & \frac{\partial g_n}{\partial x_n} \end{pmatrix} \quad \text{für} \quad g = \begin{pmatrix} g_1 \\ \vdots \\ g_n \end{pmatrix} .$$

Beweis: Sei $f(t) = g(tx + (1-t)y)$, $0 \le t \le 1$. Nach der Kettenregel ist

$$f'(t) = g'(tx + (1-t)y)(x - y) \ ,$$

also

$$\begin{aligned} \|g(x) - g(y)\| &= \|f(1) - f(0)\| = \|\int_0^1 f'(t)dt\| \\ &\le \sup_{0\le t\le 1} \|f'(t)\| \\ &= \sup_{0\le t\le 1} \|g'(tx + (1-t)y)(x - y)\| \\ &\le q\|x - y\| \ , \end{aligned}$$

weil D konvex ist.

$\square$

Satz 3.2.2 *(Kontraktionssatz, Fixpunktsatz von Banach): Sei $D \subseteq K^n$ abgeschlossen und $g : D \to D$ kontrahierend. Dann hat g in D genau einen Fixpunkt $\overline{x}$. Das Iterationsverfahren*

$$x^{k+1} = g(x^k)\ , \quad k = 0, 1, \ldots$$

konvergiert für jede Wahl von $x^0 \in D$ gegen $\overline{x}$, und es gilt mit der Lipschitz-Konstante q von g in D

$$\|x^k - \overline{x}\| \leq \frac{q^k}{1-q}\|x^1 - x^0\|\ .$$

Beweis:

1) Existenz von $\overline{x}$.

Es ist $x^k \in D$ falls $x^0 \in D$, und

$$\|x^{k+1} - x^k\| \leq q\|x^k - x^{k-1}\| \leq q^2\|x^{k-1} - x^{k-2}\| \leq ... \leq q^k\|x^1 - x^0\|\ ,$$

also für $\ell > j$

$$\begin{aligned} \|x^\ell - x^j\| &= \|\sum_{k=j}^{\ell-1}(x^{k+1} - x^k)\| \\ &\leq \sum_{k=j}^{\ell-1}\|x^{k+1} - x^k\| \\ &\leq \sum_{k-j}^{\ell-1} q^k\|x^1 - x^0\| \\ &= q^j(1 + \cdots + q^{\ell-1-j})\|x^1 - x^0\| \\ &\leq \frac{q^j}{1-q}\,\|x^1 - x^0\| \end{aligned} \tag{2.1}$$

wegen der Formel für die geometrische Reihe. Also gilt $x^\ell - x^j \to 0$ für ℓ, $j \to \infty$, d.h. (x^k) ist eine Cauchy-Folge und damit konvergent gegen ein $\overline{x} \in \mathbb{R}^n$. Da D abgeschlossen ist, ist sogar $\overline{x} \in D$, und wegen der Stetigkeit von g ist

$$\overline{x} = \lim_{k\to\infty} x^k = \lim_{k\to\infty} g(x^{k-1}) = g(\overline{x}) ,$$

also $\overline{x}$ Fixpunkt von g.

2) Eindeutigkeit

Wäre $\tilde{x}$ ein weiterer Fixpunkt von g in D, so hätten wir

$$\|\overline{x} - \tilde{x}\| = \|g(\overline{x}) - g(\tilde{x})\| \leq q\|\overline{x} - \tilde{x}\| .$$

Wegen $q < 1$ folgte $\|\overline{x} - \tilde{x}\| = 0$.

3) Fehlerabschätzung

Lassen wir in (2.1) $\ell \to \infty$ streben, so folgt

$$\|\overline{x} - x^j\| \leq \frac{q^j}{1-q}\|x^1 - x^0\| .$$

□

Beispiel: Wir wollen die Lösung von $x = \tan x$, in $[\frac{\pi}{2}, \frac{3\pi}{2}]$ durch Iteration berechnen und versuchen zunächst

$$x^{k+1} = \tan x^k .$$

Es ist $g(x) = \tan x$ und damit $g'(x) = 1/\cos^2 x \geq 1$. g ist also nicht kontrahierend. Wir müssen unsere Gleichung erst in geeignete Form bringen. Dazu schreiben wir für $x \in [\frac{\pi}{2}, \frac{3\pi}{2}]$ für $x = \tan x$

$$x = \tan(x - \pi) \quad \text{oder} \quad \arctan x = x - \pi \quad \text{oder} \quad x = \pi + \arctan x$$

und setzen $g(x) = \pi + \arctan x$. Dann ist $g'(x) = 1/(1 + x^2)$. In $D = [\frac{\pi}{2}, \frac{3\pi}{2}]$ ist dann $|g'(x)| \leq 1/(1 + \pi^2/4) < 1$, und g bildet D in sich ab. Nach Satz

3.2.1 ist g in D kontrahierend mit $q_1 = 1/(1+\pi^2/4) = 0.2884$. Also hat g in D genau einen Fixpunkt, und das Iterationsverfahren

$$x^{k+1} = \pi + \arctan x^k$$

konvergiert gegen diesen. Mit $x^0 = \pi$ erhalten wir

k	x^k	$\frac{q_1^k}{1-q_1}\lvert x^1 - x^0\rvert$	$\frac{q_2^k}{1-q_2}\lvert x^1 - x^0\rvert$	$\overline{x} - x^k$
0	3.1416	–	–	1.3518
1	4.4042	0.5117	0.1279	0.0892
2	4.4891	0.1476	0.0118	0.0043
3	4.4932	0.0426	0.0011	0.0002
4	4.4934	0.0122	0.0001	0.0000

Wir sehen, daß die Fehlerabschätzung viel zu pessimistisch ist. Man kann auch $D = [\pi, \frac{3\pi}{2}]$ wählen mit $q_2 = 1/(1+\pi^2) = 0.092$ und bekommt dann bessere Abschätzungen.

Welche Fixpunkte von g sind durch das Iterationsverfahren $x^{k+1} = g(x^k)$ berechenbar? - Durch Skizzen im $\mathbb{R}^1$ kommt man zu der Vermutung, daß dies die “anziehenden” Fixpunkt sind, d.h. diejenigen mit $|g'(\overline{x})| < 1$. Dies ist der Inhalt des nächsten Satzes.

Satz 3.2.3 *(Lokaler Konvergenzsatz): Die Abbildung $g : K^n \to K^n$ besitze einen Fixpunkt $\overline{x}$, und es gebe eine Umgebung von $\overline{x}$, in der g kontrahierend ist. Dann gibt es eine Umgebung $U(\overline{x})$, so daß das Iterationsverfahren $x^{k+1} = g(x^k)$ für jedes $x^0 \in U(\overline{x})$ gegen $\overline{x}$ konvergiert.*

Beweis: Wir können annehmen, daß g in $D = \{x \in K^n : \|x - \overline{x}\| \le r\}$ mit einem, $r > 0$ kontrahierend ist, und zwar mit der Lipschitz-Konstanten $q < 1$. Dann ist für $x \in D$

$$\|g(x) - \overline{x}\| = \|g(x) - g(\overline{x})\| \le q\|x - \overline{x}\| < qr\ ,$$

also auch $g(x) \in D$. g bildet also die abgeschlossene Menge D in sich ab und ist dort kontrahierend. Nach Satz 3.2.2 konvergiert das Iterationsverfahren für $x^0 \in D$.

$\square$

Bemerkungen:

1) Sei $K = \mathbb{R}$. Die Bedingung des Satzes III.2.3 ist erfüllt, wenn g in $\overline{x}$ stetig differenzierbar ist und $\rho(g'(\overline{x})) < 1$ ist. Dann gibt es nämlich nach Satz I.3.2 eine Norm $\|\cdot\|$ in $\mathbb{R}^n$ mit $\|g'(\overline{x})\| < 1$ für $x = \overline{x}$. Wegen der Stetigkeit von g' gilt dies dann auch in einer konvexen Umgebung von $\overline{x}$. Nach Satz III.2.1 ist g in dieser Umgebung kontrahierend.

2) Für die Konvergenzgeschwindigkeit des Iterationsverfahrens ist offenbar die Lipschitz-Konstante q entscheidend:

$$\|x^{k+1} - \overline{x}\| = \|g(x^k) - g(\overline{x})\| \leq q\|x^k - \overline{x}\| \ .$$

Der Fehler vermindert sich also in jedem Schritt (mindestens) um den Faktor $q < 1$. Wir sprechen von linearer Konvergenz. Nach Satz III.2.3 und Bemerkung 1 ist für die Konvergenzgeschwindigkeit die Zahl $\rho(g'(\overline{x}))$ entscheidend. Gilt hingegen für ein Iterationsverfahren

$$\|x^{k+1} - \overline{x}\| \leq C\|x^k - \overline{x}\|^p$$

mit einer Zahl $p > 1$, so sprechen wir von Konvergenz (mindestens) der Ordnung p. Für $p = 2$ sprechen wir von quadratischer, für $p = 3$ von kubischer Konvergenz. Quadratische Konvergenz ist dadurch gekennzeichnet, daß sich die Anzahl der korrekten Dezimalen von x^k bei jedem Schritt verdoppelt.

3.3 Das Newton-Verfahren

Zu lösen sei das nichtlineare System $f(x) = 0$, $f : D \to \mathbb{R}^n$, $D \subseteq \mathbb{R}^n$. Die Konvergenzgeschwindigkeit des Iterationsverfahrens $x^{k+1} = g(x^k)$ wird nach Bemerkung 2 aus dem vorigen Paragraphen durch die Zahl $\rho(g'(\overline{x}))$ bestimmt. Bei der Umwandlung einer Gleichung der Form $f(x) = 0$ in eine Fixpunktgleichung $x = g(x)$ versuchen wir daher $g'(\overline{x}) = 0$ zu erreichen. Wir machen für g den Ansatz

$$g(x) = x + A(x)f(x)$$

mit einer invertierbaren Matrix $A = (a_{i,j})$, d.h.

$$g_i(x) = x_i + \sum_{j=1}^{n} a_{i,j}(x)f_j(x)\ .$$

Wir berechnen die Jacobi-Matrix $g'(x)$. Es ist

$$\frac{\partial g_i}{\partial x_\ell} = \delta_{i,\ell} + \sum_{j=1}^{n} \frac{\partial a_{i,j}}{\partial x_\ell} f_j + \sum_{j=1}^{n} a_{i,j} \frac{\partial f_j}{\partial x_\ell}\ .$$

Für $x = \overline{x}$ ist $f(\overline{x}) = 0$, und wir können diese Gleichungen zusammenfassen zu

$$g'(\overline{x}) = I + A(\overline{x})f'(\overline{x})\ .$$

Um $g'(\overline{x}) = 0$ zu erreichen, brauchen wir also nur

$$A(x) = -(f'(x))^{-1}$$

zu wählen. Damit wird

$$g(x) = x - (f'(x))^{-1}f(x)\ , \tag{3.1}$$

und das Iterationsverfahren zur Lösung von $f(x) = 0$ lautet

$$x^{k+1} = x^k - (f'(x^k))^{-1}f(x^k)\ . \tag{3.2}$$

In dieser Form verlangt das Verfahren die Inversion von $f'(x^k)$. Dies ist für große n unzweckmäßig. Man schreibt daher

$$f'(x^k)(x^{k+1} - x^k) + f(x^k) = 0\ . \tag{3.3}$$

Dies ist das Newton-Verfahren zur Lösung von $f(x) = 0$. Man hätte es auch einfacher durch Linearisierung herleiten können. Ist x^k eine Näherung für die gesuchte Lösung $\overline{x}$, so hat man für $f \in C^2(D)$

$$f(x) = f(x^k) + f'(x^k)(x - x^k) + \text{ O } (\|x - x^k\|^2) \ .$$

Man vernachlässigt nun $\text{O}(\|x - x^k\|^2)$ und nimmt als neue Näherung x^{k+1} für $\overline{x}$ die Lösung von

$$0 = f(x^k) + f'(x^k)(x - x^k) \ .$$

Dies ist genau (3.3). Im $\mathbb{R}^1$ ersetzt das Newton-Verfahren also die Kurve $y = f(x)$ durch die Tangente in x^k und berechnet x^{k+1} als Nullstelle der Tangente.

Beispiel: Wir lösen in $\mathbb{R}^1$ die Gleichung $x^2 - 2 = 0$, berechnen also $\overline{x} = \sqrt{2}$. Es ist

$$f(x) = x^2 - 2 \ , \quad q(x) = x - \frac{f(x)}{f'(x)} = x - \frac{x^2 - 2}{2x} = \frac{1}{2}(x + \frac{2}{x}) \ .$$

Mit $x^0 = 1$ erhalten wir

k	x_k	Anzahl der korrekten Dezimalen
0	1	1
1	$1.\underline{5}$	1
2	$1.41\underline{7}$	3
3	$1.41421\underline{6}$	6
4	$1.41421356237\underline{5}$	12

Satz 3.3.1 *Die Abbildung $f : \mathbb{R}^n \to \mathbb{R}^n$ besitze die Nullstelle $\overline{x}$, sei in einer Umgebung von $\overline{x}$ zweimal stetig differenzierbar, und $f'(\overline{x})$ sei invertierbar. Dann gibt es eine Umgebung D von $\overline{x}$, so daß das Newton-Verfahren für alle $x^0 \in D$ quadratisch gegen $\overline{x}$ konvergiert.*

Beweis: Sei $g(x) = x - (f'(x))^{-1} f(x)$. Wegen $g'(\overline{x}) = 0$ gibt es eine Umgebung D von $\overline{x}$ mit $\|g'(x)\| \leq 1/2$ für $x \in D$. Nach Satz III.2.3 konvergiert das Iterationsverfahren $x^{k+1} = g(x^k)$, also das Newton-Verfahren, gegen $\overline{x}$, wenn nur $x^0 \in D$.

Zum Nachweis der quadratischen Konvergenz bilden wir

$$x^{k+1} - \overline{x} = x^k - \overline{x} - (f'(x^k))^{-1}(f(x^k) - f(\overline{x})) \; .$$

Der Satz von Taylor liefert

$$f(x^k) - f(\overline{x}) = f'(x^k)(x^k - \overline{x}) + \varepsilon(\overline{x} - x^k) \; ,$$
$$\|\varepsilon(\overline{x} - x^k)\| \leq M\|\overline{x} - x^k\|^2$$

mit einer Konstanten M, welche die zweiten Ableitungen von f enthält. Dazu setzen wir D so klein voraus, daß $f \in C^2(D)$. Es folgt

$$\begin{aligned} x^{k+1} - \overline{x} &= x^k - \overline{x} - (x^k - \overline{x}) - (f'(x^k))^{-1}\varepsilon(\overline{x} - x^k) \\ &= -f'(x^k)^{-1}\varepsilon(\overline{x} - x^k) \; . \end{aligned}$$

Sei nun N eine Konstante mit $\|(f'(x^k))^{-1}\| \leq N$. Dann gilt

$$\|x^{k+1} - \overline{x}\| \leq NM\|x^k - \overline{x}\|^2 \; ,$$

und dies bedeutet quadratische Konvergenz.

□

Was passiert, wenn $f'(\overline{x})$ nicht invertierbar ist? - Wir betrachten den Fall $n = 1$. Sei also $f(\overline{x}) = 0,\; f \in C^2,\; f'(\overline{x}) = 0$, aber $f''(\overline{x}) \neq 0$. Dann ist

$$f(x) = (x - \overline{x})^2 p(x) \; , \quad p(\overline{x}) \neq 0$$

mit $p \in C^2$. Wir berechnen

$$f'(x) = 2(x-\overline{x})p(x)+(x-\overline{x})^2p'(x), f''(x) = 2p(x)+4(x-\overline{x})p'(x)+(x-\overline{x})^2p''(x)$$

und erhalten für $g(x) = x - f(x)/f'(x)$

$$\begin{aligned} g'(x) &= 1 - \frac{f'(x)}{f'(x)} + \frac{f(x)f''(x)}{f'^2(x)} = \frac{(x-\overline{x})^2p(x)(2p(x) + \mathrm{O}(x-\overline{x}))}{4(x-\overline{x})^2(p(x) + \mathrm{O}(x-\overline{x}))^2} \\ &= \frac{1}{2}(1 + \mathrm{O}(x - \overline{x})) \; . \end{aligned}$$

Also ist $g'(\overline{x}) = \frac{1}{2}$, und nach Satz III.2.3 folgt lineare Konvergenz. Für $f'(\overline{x}) = 0$ konvergiert das Newton-Verfahren also immer noch, aber die quadratische Konvergenz geht verloren.

Der Rechenaufwand für das Newton-Verfahren wird im wesentlichen durch die Berechnung von $f'(x)$ bestimmt. Es gibt verschiedene Varianten des Newton-Verfahrens, die diesen Aufwand reduzieren.

1. Das vereinfachte Newton-Verfahren

Hier ersetzt man einfach $\left(f'(x^k)\right)^{-1}$ durch $\left(f'(x^0)\right)^{-1}$, iteriert also gemäß

$$f'(x^0)(x^{k+1} - x^k) = -f(x^k)\ .$$

Man hat immer noch lokale Konvergenz, aber die quadratische Konvergenz geht verloren.

2. Das Sekanten-Verfahren ("Regula falsi") $(n = 1)$

Hier ersetzt man die Tangente des Newton-Verfahrens durch die Sekante in den Punkten x^k, x^{k-1}, also

$$y = f(x^k) + (x - x^k)\ \frac{f(x^k) - f(x^{k-1})}{x^k - x^{k-1}}\ ,$$

und nimmt als neue Näherung x^{k+1} die Nullstelle dieser Sekante, also

$$x^{k+1} = x^k - \left(\frac{f(x^k) - f(x^{k-1})}{x^k - x^{k-1}}\right)^{-1} f(x^k)\ .$$

Mit anderen Worten: Man ersetzt $f'(x^k)$ durch den Differenzenquotienten in x^k, x^{k-1}. Dies ist auch in $\mathbb{C}$ sinnvoll.

Satz 3.3.2 *Sei $f : \mathbb{R} \to \mathbb{R}$ in einer Umgebung der Nullstelle $\overline{x}$ zweimal stetig differenzierbar, und sei $f'(\overline{x}) \neq 0$. Dann gibt es eine Umgebung D von $\overline{x}$, so daß das Sekanten-Verfahren für jede Wahl von x^0, x^1 in D gegen $\overline{x}$ konvergiert, und zwar gilt $\|\overline{x} - x^k\| \leq c_k$, wobei $c_k \to 0$ mit der Konvergenzordnung $(1 + \sqrt{5})/2 = 1.618$.*

Beweis:

(a) Es gibt eine Konstante C, so daß

$$|x^{k+1} - \overline{x}| \leq C|x^k - \overline{x}||x^{k-1} - \overline{x}| \;, \quad k = 1, 2, \dots \;.$$

Hierzu schreiben wir

$$\begin{aligned} x^{k+1} - \overline{x} &= x^k - \overline{x} - \frac{x^k - x^{k-1}}{f(x^k) - f(x^{k-1})}\, f(x^k) \\ &= (x^k - \overline{x})\, \frac{\dfrac{f(x^k) - f(x^{k-1})}{x^k - x^{k-1}} - \dfrac{f(x^k) - f(\overline{x})}{x^k - \overline{x}}}{\dfrac{f(x^k) - f(x^{k-1})}{x^k - x^{k-1}}} \\ &= (x^k - \overline{x})\, \frac{\int_0^1 \{f'(x^{k-1} + t(x^k - x^{k-1})) - f'(\overline{x} + t(x^k - \overline{x}))\} dt}{f'(\xi^k)} \end{aligned}$$

mit ξ^k zwischen x^k und x^{k-1}. In einer Umgebung D von $\overline{x}$ ist

$$|f'(x)| \geq m > 0 \;, \quad |f''(x)| \leq M$$

und daher

$$|x^{k+1} - \overline{x}| \leq |x^k - \overline{x}|\, \frac{M|x^{k-1} - \overline{x}|}{m} \;,$$

falls x_k, $x_{k-1} \in D$. Ist $D = [\overline{x} - \varepsilon, \overline{x} + \varepsilon]$ und x^0, $x^1 \in D$, so folgt also $|x_2 - \overline{x}| \leq \varepsilon^2 \frac{M}{m}$, und für $\varepsilon \frac{M}{m} \leq 1$ ist auch $x_2 \in D$. Damit bleiben alle x^k in D, und (a) ist gezeigt.

(b) Wir setzen $e^{f_k} = C|x^k - \overline{x}|$. Dann wird

$$f_{k+1} \leq f_k + f_{k-1} \;.$$

Sei $F_0 = F_1 = 1$ und $F_{k+1} = F_k + F_{k-1}$. Ist $|f_0|$, $|f_1| \leq 1$, so gilt offenbar $f_k \leq F_k$, $k = 0, 1, \dots$. Die F_k kann man leicht berechnen. Mit $\tau = \frac{1}{2}(1 + \sqrt{5})$ ist

$$F_k = c_1 \tau^k + c_2 (-\tau)^{-k} \;,$$

$$c_1 = \frac{\tau^2}{1 + \tau^2} \quad , \quad c_2 = \frac{1}{1 + \tau^2} \;.$$

Wir setzen nun $\varepsilon_k = e^{F_k}$. Dann ist

$$\frac{\varepsilon_k}{\varepsilon_{k-1}^{\tau}} = e^{F_k - \tau F_{k-1}} = e^{c_2((-\tau)^{-k} - \tau(-\tau)^{-k+1})}$$

und dies strebt wegen $\tau > 1$ für $k \to \infty$ gegen 1. Also gilt $\varepsilon_k \leq c\varepsilon_{k-1}^{\tau}$ für ein geeignetes $c > 0$, und

$$|x^k - \overline{x}| \leq \frac{1}{c} e^{f_k} \leq \frac{1}{c} e^{F_k} = \frac{1}{c} \varepsilon_k \ .$$

Die Behauptung des Satzes folgt mit $c_k = \varepsilon_k / c$.

$\square$

Kapitel 4

Iterationsverfahren für lineare Gleichungssysteme

4.1 Gesamt- und Einzelschrittverfahren

Das Standardverfahren zur Lösung linearer Gleichungssysteme ist das Eliminationsverfahren. Bei sehr großen Systemen mit spezieller Struktur können iterative Verfahren aber günstiger sein. Dies trifft insbesondere zu auf lineare Systeme, deren Matrizen nur sehr wenige von Null verschiedene Elemente haben (dünnbesetzte Matrizen).

Sei $Ax = b$ zu lösen. Sei $A = D + L + R$ mit

$$D = \begin{pmatrix} a_{1,1} & & \mathbf{O} \\ & \ddots & \\ \mathbf{O} & & a_{n,n} \end{pmatrix}, \; L = \begin{pmatrix} 0 & & & \mathbf{O} \\ a_{1,1} & \ddots & & \\ \vdots & & & \\ a_{n,1} & \dots & a_{n,n-1} & 0 \end{pmatrix}, \; R = \begin{pmatrix} 0 & a_{1,2} & \dots & a_{1,n} \\ & \ddots & & \vdots \\ & & & a_{n-1,n} \\ \mathbf{O} & & & 0 \end{pmatrix}.$$

Das Gesamtschritt- oder Jacobi-Verfahren zur Lösung von $(D + L + R)x = b$ lautet

$$Dx^{k+1} + Lx^k + Rx^k = b\,,$$

während das Einzelschritt- oder Gauß-Seidel-Verfahren gemäß

$$(D + L)x^{k+1} + Rx^k = b$$

iteriert. Elementweise lauten Gesamt- bzw. Einzelschrittverfahren

$$x_i^{k+1} = \left(b_i - \sum_{j\neq i} a_{i,j}x_j^k\right)/a_{i,i} ,$$

$$x_i^{k+1} = \left(b_i - \sum_{j=1}^{i-1} a_{i,j}x_j^{k+1} - \sum_{j=1+1}^{n} a_{i,j}x_j^k\right)/a_{i,i} .$$

Der rechentechnische Unterschied zwischen den beiden Verfahren zeigt sich am deutlichsten bei der Programmierung

$GS(x,n)$ /⋆ Führt einen Gesamtschritt durch ⋆/
{ for $i = 1,\ldots,n$
  $x_i^1 = (b_i - \sum_{j\neq i} a_{i,j}x_j)/a_{i,i}$;
  $x = x^1$;
}
$ES(x,n)$ /⋆ Führt einen Einzelschritt durch ⋆/
{ for $i = 1,\ldots,n$
  $x_i = (b_i - \sum_{j\neq i} a_{i,j}x_j)/a_{i,i}$;
}

Die Iterationsvorschrift des Einzelschrittverfahrens wird also durch sukzessives Überschreiben ausgeführt. Man braucht also nicht zwei Vektoren x, x^1 zu halten, sondern kommt mit einem aus. Da x häufig sehr groß ist, ist dies von Vorteil. Darüber hinaus werden wir sehen, daß das Einzelschrittverfahren häufig schneller konvergiert als das Gesamtschrittverfahren.

Beispiel: Wir wollen das Dirichlet-Problem

$$-\Delta u = f \quad \text{in } \Omega , \quad u = 0 \quad \text{auf } \partial\Omega$$

für das Quadrat $\Omega = (0,1)^2$ in $\mathbb{R}^2$ lösen. Hier ist $\Delta = \partial^2/\partial x^2 + \partial^2/\partial y^2$ der Laplace-Operator. Man überdeckt Ω durch ein Gitter mit Schrittweite $h = \frac{1}{n}$ und ersetzt die Differentialgleichung im Gitterpunkt $\begin{pmatrix} i \\ j \end{pmatrix} h$ durch

$$4u_{i,j} - (u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) = h^2 f_{i,j} , \quad i,j = 1,\ldots,n-1 .$$

Dies ist ein lineares System von $(n-1)^2$ Gleichungen in ebensovielen Unbekannten $u_{i,j}$; die $u_{i,j}$ mit $i,j \in \{0,n\}$ werden Null gesetzt. Ist h hinreichend klein, so hofft man, daß $u_{i,j}$ eine gute Näherung für $u\left(\begin{pmatrix} i \\ j \end{pmatrix} h\right)$ ist.

Ein Einzelschritt lautet nun einfach

$$\begin{aligned} \text{for } i &= 1,\ldots,n-1 \quad \text{for } \ j = 1,\ldots,n-1 \\ u_{i,j} &= (h^2 f_{i,j} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1})/4 \ ; \end{aligned}$$

Beide Verfahren kann man beschleunigen durch Einführung eines "Relaxationsparameters" ω. Für das Gesamtschrittverfahren setzt man

$$x_i^{k+1} = (1-\omega)x_i^k + \omega(b_i - \sum_{j \neq i} a_{i,j} x_j^k)/a_{i,i} \ ,$$

und für das Einzelschrittverfahren

$$x_i^{k+1} = (1-\omega)x_i^k + \omega \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{k+1} - \sum_{j=i+1}^{n} a_{i,j} x_j^k \right) /a_{i,i} \ .$$

Man bildet also gewichtete Summen der k-ten und der $k+1$-ten Näherung. Das letzte Verfahren heißt SOR (successive overrelaxation) und spielt eine wichtige Rolle. Für die Programmierung von SOR gilt das über das Einzelschrittverfahren Gesagte.

In Matrizenschreibweise haben alle diese Verfahren die Form

$$x^{k+1} = Bx^k + c \ .$$

Dabei ist für das Gesamtschrittverfahren

$$B = -D^{-1}(L+R) \ , \quad c = D^{-1}b \ ,$$

für das Einzelschrittverfahren

$$B = -(D+L)^{-1}R \ , \quad c = (D+L)^{-1}b$$

und für das SOR-Verfahren

$$B = (D+\omega L)^{-1}((1-\omega)D - \omega R) \ , \quad c = \omega(D+\omega L)^{-1}b \ .$$

4.2 Konvergenz

Alle unsere Konvergenzaussagen werden auf folgendem Satz beruhen:

Satz 4.2.1 *Das Iterationsverfahren*

$$x^{k+1} = Bx^k + c$$

konvergiert genau dann für jede Wahl von x^0 und c, wenn $\rho(B) < 1$. In diesem Fall konvergiert es gegen die eindeutig bestimmte Lösung $\overline{x}$ von $\overline{x} = B\overline{x} + c$.

Beweis: Ist $\rho(B) < 1$, so gibt es nach Satz 1.6.2 eine Norm $\| \cdot \|$, so daß $\|B\| < 1$. Damit ist $g(x) = Bx + c$ kontrahierend im ganzen Raum. Konvergenz gegen $\overline{x}$ folgt dann aus Satz 3.2.2.

Sei umgekehrt das Verfahren für alle x^0, c konvergent. Wir wählen für x^0, c den Eigenvektor y von B mit Eigenwert λ. Dann ist

$$x^k = (1 + \lambda + \cdots + \lambda^k)y \ .$$

Dies ist nur konvergent für $|\lambda| < 1$. Also ist $\rho(B) < 1$.

$\square$

Definition 4.2.1 *Eine (n, n)-Matrix A über K erfüllt das starke Zeilensummenkriterium, wenn $|a_{i,i}| > \sum\limits_{j \neq i} |a_{i,j}|, \quad i = 1, \ldots, n$. Sie erfüllt das schwache Zeilensummenkriterium, wenn diese Ungleichungen mit mindestens einer Ausnahme im schwachen Sinne (d.h. mit $\geq$ an Stelle von $>$) gelten.*

A heißt reduzibel, wenn man $\{1, \ldots, n\}$ so in zwei disjunkte nichtleere Mengen I, J aufteilen kann, daß $a_{i,j} = 0$ für $(i, j) \in I \times J$. Andernfalls heißt A irreduzibel.

Ist A reduzibel, so kann man A durch Zeilen- und Spaltenvertauschungen in die Form

$$A = \begin{pmatrix} A_1 & 0 \\ A_3 & A_2 \end{pmatrix}$$

bringen. Das Gleichungssystem $Ax = b$ zerfällt dann in zwei kleinere Systeme mit den Matrizen A_1, A_2.

Satz 4.2.2 *Für die (n,n)-Matrix A sei eine der folgenden Bedingungen erfüllt:*

(a) A erfülle das starke Zeilensummenkriterium.

(b) A erfülle das schwache Zeilensummenkriterium und sei irreduzibel.

Dann konvergiert das Gesamtschrittverfahren zur Lösung von $Ax = b$ für jedes b und x^0 gegen die eindeutig bestimmte Lösung $\overline{x}$ von $A\overline{x} = b$.

Beweis: Wir zeigen, daß $\rho(D^{-1}(L+R)) < 1$, wo $A = D + L + R$. Unter der Voraussetzung (a) ist

$$\|D^{-1}(L+R)\|_\infty = \max_{i=1}^{n} \frac{1}{|a_{i,i}|} \sum_{j \neq i} |a_{i,j}| < 1 \ ,$$

also in der Tat $\rho(D^{-1}(L+R)) < 1$. Unter der Voraussetzung (b) folgt nur $\rho(D^{-1}(L+R)) \leq 1$. Es genügt daher zu zeigen, daß $D^{-1}(L+R)$ keinen Eigenwert λ mit $|\lambda| = 1$ hat. Wäre λ ein solcher und x ein zugehöriger Eigenvektor mit $\|x\|_\infty = 1$, so hätten wir für alle $i \in I = \{i : |x_i| = 1\}$

$$1 = |x_i| = |\lambda||x_i| = |(D^{-1}(L+R)x)_i| = \frac{1}{|a_{i,i}|} \left| \sum_{j \neq i} a_{i,j} x_j \right| \leq 1 \ ,$$

also

$$\left| \sum_{j \neq i} a_{i,j} x_j \right| = |a_{i,i}| \ , \quad i \in I \ .$$

Wegen des schwachen Zeilensummenkriteriums kann $J = \{1, \ldots, n\} \setminus I$ nicht leer sein. Da A irreduzibel ist, gibt es $(i_0, j_0) \in I \times J$ mit $a_{i_0,j_0} \neq 0$. Dann ist

$$
\begin{aligned}
1 = |\lambda||x_{i_0}| &= |(D^{-1}(L+R)x)_{i_0}| = \frac{1}{|a_{i_0,i_0}|}\left|\sum_{j\neq i_0} a_{i_0,j}x_j\right| \\
&< \frac{1}{|a_{i_0,i_0}|}\sum_{j\neq i_0} |a_{i_0,j}| \leq 1 \ .
\end{aligned}
$$

Dieser Widerspruch zeigt, daß es keinen solchen Eigenwert λ geben kann.

$\square$

Satz 4.2.3 *Die (n,n)-Matrix A sei positiv definit. Dann konvergiert das SOR-Verfahren zur Lösung von $Ax = b$ für jede Wahl von x^0 und b, wenn $0 < \omega < 2$.*

Beweis: Wir haben zu zeigen, daß für die SOR-Matrix $B_\omega = (D+\omega L)^{-1}((1-\omega)D - \omega R)$ der Spektralradius $\rho(B_\omega) < 1$ ist für $0 < \omega < 2$. Sei x ein Eigenvektor von B_ω zum Eigenwert λ mit $|\lambda| = \rho(B_\omega)$, also

$$((1-\omega)D - \omega R)x = \lambda(D+\omega L)x \ .$$

Inneres Produkt mit x ergibt

$$(1-\omega)(Dx,x) - \omega(Rx,x) = \lambda((Dx,x) + \omega(Lx,x)) \ .$$

Wir setzen $d = (Dx,x)$, $\ell = (Lx,x)$. Weil A hermitesch ist, gilt $(Rx,x) = (L^*x,x) = (x,Lx) = \overline{\ell}$, also

$$(1-\omega)d - \omega\overline{\ell} = \lambda(d+\omega\ell)$$

oder

$$\lambda = \frac{(1-\omega)d - \omega\overline{\ell}}{d+\omega\ell} \ .$$

Mit $\ell = \alpha + i\beta$, α, $\beta \in \mathbb{R}$ bekommen wir, weil $d > 0$,

$$|\lambda|^2 = \frac{((1-\omega)d - \omega\alpha)^2 + \omega^2\beta^2}{(d+\omega\alpha)^2 + \omega^2\beta^2} \ .$$

Es ist also genau dann $|\lambda| < 1$, wenn

$$|(1-\omega)d - \omega\alpha| < |d + \omega\alpha| \ .$$

Mit $\alpha' = \alpha/d$ lautet dies

$$|1 - \omega - \omega\alpha'| < |1 + \omega\alpha'| \ . \tag{2.1}$$

Da A positiv definit ist, gilt

$$0 < (Ax, x) = d + \ell + \overline{\ell} = d + 2\alpha = d(1 + 2\alpha') \ ,$$

also $\alpha' > -1/2$. Für $0 < \omega < 2$ können wir die Betragsstriche bei $|1 + \omega\alpha'|$ in (2.1) weglassen und erhalten

$$|1 - \omega - \omega\alpha'| < 1 + \omega\alpha'$$

als Bedingung für $|\lambda| < 1$. Dies ist aber für $0 < \omega < 2$ richtig.

□

Die Bedingung an ω kann man nicht abschwächen. Es gilt nämlich

Satz 4.2.4 *Sei B_ω die SOR-Matrix für eine beliebige Matrix mit nichtverschwindenden Diagonalelementen. Dann gilt*

$$\rho(B_\omega) \geq |\omega - 1| \ .$$

Beweis: Es ist

$$B_\omega = (I + \omega D^{-1} L)^{-1}((1-\omega)I - \omega D^{-1} R) \ .$$

Die Diagonale von B_ω ist also $(1-\omega)I$. Sind $\lambda_1, \ldots, \lambda_n$ die Eigenwerte von A (nicht notwendig verschieden), so gilt

$$\rho(B_\omega)^n \geq |\lambda_1 \cdots \lambda_n| = |1 - \omega|^n \ ,$$

und daraus folgt die Behauptung.

□

4.3 Das Verfahren der konjugierten Gradienten (CG)

Ein Problem des SOR-Verfahrens liegt in der Bestimmung des optimalen Wertes von ω. Das CG-Verfahren kommt ohne einen solchen Parameter aus und hat trotzdem gute Konvergenzeigenschaften.

Sei A eine reelle positiv definite (n,n)-Matrix. Zu lösen sei $Ax = b$. Wir setzen

$$f(x) = \frac{1}{2}(x, Ax) - (x, b) \; .$$

Satz 4.3.1 *$\overline{x}$ ist genau dann Lösung von $A\overline{x} = b$, wenn $f(\overline{x}) \leq f(x)$ für alle $x \in \mathbb{R}^n$.*

Beweis: Ist $\overline{x}$ die Lösung, so ist

$$\frac{1}{2}(A(x-\overline{x}), (x-\overline{x})) = \frac{1}{2}(Ax, x) - (x, b) + \frac{1}{2}(A\overline{x}, \overline{x}) \; .$$

$\overline{x}$ ist also der eindeutig bestimmte Vektor, welcher $f(x)$ minimiert.

$\square$

Anstelle einer Lösung von $Ax = b$ können wir also genausogut das Minimum von $f(x)$ suchen. Sei x^0 eine Ausgangsnäherung für $\overline{x}$. Wir suchen eine bessere Näherung für $\overline{x}$ in der Form $x^0 + \alpha d^0$, wo $d^0 \neq 0$ eine "Suchrichtung" ist. Wir bestimmen α so, daß $f(x^0 + \alpha d^0)$ minimal ist. Wegen $f'(x) = Ax - b$ ist

$$\begin{aligned}
\frac{d}{d\alpha}\, f(x^0 + \alpha d^0) &= (f'(x^0 + \alpha d^0), d^0) \\
&= (A(x^0 + \alpha d^0) - b, d^0) \\
&= (r^0, d^0) + \alpha(Ad^0, d^0) \; , \\
\frac{d^2}{d\alpha^2}\, f(x^0 + \alpha d^0) &= (Ad^0, d^0) > 0 \; ,
\end{aligned}$$

wobei wir hier und unten $r^k = Ax^k - b$ setzen. Entlang $x^0 + \alpha d^0$ ist also f genau dann minimal, wenn die erste Ableitung von f verschwindet. Dies ist für

$$\alpha_0 = -\frac{(r^0, d^0)}{(Ad^0, d^0)}$$

der Fall, und wir setzen $x^1 = x^0 + \alpha_0 d^0$. Geometrisch bedeutet diese Wahl, daß $r^1 \perp d^0$, und d^0 liegt im Tangentialraum des Ellipsoids $f(x) = f(x^1)$ im Punkte x^1.

Wie wählt man nun d^0? Eine naheliegende Wahl ist $d^0 = -r^0$, d.h. man läuft in Richtung des steilsten Abstiegs von f. Dies führt zu den Verfahren des steilsten Abstiegs (steepest descent) oder Gradientenverfahren:

Initialisierung	:	Wähle x^0, $r^0 = Ax^0 - b,\ \ d^0 = -r^0$.
Für $k = 0, 1, 2, \ldots$	:	$z = Ad^k$
		$\alpha_k = \dfrac{(r^k, d^k)}{(z, d^k)}$
		$x^{k+1} = x^k + \alpha_k d^k \quad , \quad d^{k+1} = d^k - \alpha_k z\ .$

Die Suchrichtung beim k-ten Schritt ist $d^k = -r^k$.

Wir haben gesehen, daß für das Gradientenverfahren $r^{k+1} \perp d^k$ ist, also $d^{k+1} \perp d^k$. Dies bedeutet, daß x^k sich $\overline{x}$ auf einem Zick-Zack-Kurs annähert. Dies läßt für die Konvergenzgeschwindigkeit nichts Gutes hoffen.

Beim CG-Verfahren ersetzt man die eindimensionale Minimierung durch eine zweidimensionale. Sind x^0, x^1 nach den Gradientenverfahren berechnet, so setzt man

$$x^2 = x^1 - \alpha r^1 + \beta d^0$$

und bestimmt α, β so, daß $f(x^2)$ minimal ist. Die Bedingungen für ein Minimum sind

$$\begin{aligned} \frac{\partial f}{\partial \alpha}(x^2) &= (Ax^2 - b, r^1) = 0 \\ \frac{\partial f}{\partial \beta}(x^2) &= (Ax^2 - b, d^0) = 0\ . \end{aligned}$$

Auflösen dieser Gleichungen nach α, β liefert x^2. Wir benutzen nur die zweite Gleichung, welche wir in der Form

$$(r^1 - \alpha A r^1 + \beta A d^0, d^0) = 0$$

schreiben können. Wegen $(r^1, d^0) = 0$ folgt

$$\beta = \alpha \beta_0 \quad , \quad \beta_0 = \frac{(Ar^1, d^0)}{(Ad^0, d^0)}$$

und damit

$$x^2 = x^1 + \alpha d^1 \quad , \quad d^1 = -r^1 + \beta_0 d^0 \ .$$

Nun können wir α durch eine eindimensionale Minimierung bestimmen. Das Resultat α_1 kennen wir schon vom Gradientenverfahren, nämlich

$$\alpha_1 = -\frac{(r^1, d^1)}{(d^1, Ad^1)} \ .$$

Damit ist x^2 bestimmt, allerdings in einer sehr unpraktischen Form: Die Berechnung von x^2 erfordert drei Anwendungen der Matrix A auf einen Vektor. Wir wollen mit einer Anwendung auskommen. Dazu schreiben wir β_0 in der Form

$$\beta_0 = \frac{(r^1, Ad^0)}{(d^0, Ad^0)} = \frac{(r^1, (r^1 - r^0)/\alpha_0)}{(d^0, Ad^0)} = \frac{(r^1, r^1)}{(r^0, r^0)} \ ,$$

wobei, wie beim Gradientenverfahren, $\alpha_0 = (r^0, r^0)/(d^0, Ad^0)$. Für α_1 haben wir die α_0 entsprechende Form

$$\alpha_1 = -\frac{(r^1, -r^1 + \beta_0 d^0)}{(d^1, Ad^1)} = \frac{(r^1, r^1)}{(d^1, Ad^1)} \ ,$$

denn $(r^1, d^0) = 0$. Damit lautet das CG-Verfahren:

Initialisierung	:	Wähle x^0, $r^0 = Ax^0 - b$, $d^0 = -r^0$
Für $k = 0, 1, \ldots$	:	$z = Ad^k$
		$\alpha_k = \dfrac{(r^k, r^k)}{(d^k, z)}$
		$x^{k+1} = x^k + \alpha_k d^k \ , \ r^{k+1} = r^k + \alpha_k z$
		$\beta_k = \dfrac{(r^{k+1}, r^{k+1})}{(r^k, r^k)}$
		$d^{k+1} = -r^{k+1} + \beta_k d^k \ .$

Für $k = 0$ macht also das CG-Verfahren einen Gradientenschritt. Die weiteren Schritte führen wie das Gradientenverfahren eine eindimensionale Minimierung durch, aber nicht entlang des steilsten Abstiegs $-r^k$, sondern entlang der Richtung d^k. Das Verfahren bricht ab, wenn $r^k = 0$. In diesem Fall ist $x^k = \overline{x}$.

Satz 4.3.2 *Das CG-Verfahren bricht nach höchstens n Schritten mit $r^k = 0$ ab. Solange es nicht abbricht, gilt*

(a) $(d^i, r^k) = 0$ *für* $i < k$,

(b) $(r^k, d^k) = -\|r^k\|^2$,

(c) $(r^i, r^k) = 0$ *für* $i < k$,

(d) $(d^i, Ad^k) = 0$ *für* $i < k$,

(e) $sp\ \{d^0, \ldots, d^k\} = sp\ \{r^0, \ldots, r^k\}$.

Beweis: Wir beweisen (a) - (e) durch Induktion nach k. Die Aussage über den Abbruch folgt dann aus (c), weil es nicht mehr als n linear unabhängige Vektoren r^k gibt. Solange $r^k \neq 0$ ist, ist nach (b) auch $d^k \neq 0$. Das Verfahren bricht also nur mit $r^k = 0$ ab. Für $k = 0$ sind (a) - (e) trivial. Sei (a) - (e) richtig für ein $k \geq 0$. Wir zeigen, daß (a) - (e) dann auch für $k+1$ richtig ist, wenn nur $r^{k+1} \neq 0$.

(a) Es ist für $i < k$

$$(d^i, r^{k+1}) = (d^i, r^k + \alpha_k Ad^k) = 0$$

wegen (a) und (d) für k. Für $i = k$ ist

$$(d^k, r^{k+1}) = 0 \ ,$$

weil x^{k+1} durch Minimierung entlang d^k entsteht.

(b) Es ist

$$(r^{k+1}, d^{k+1}) = (r^{k+1}, -r^{k+1} + \beta_k d^k) = -\|r^{k+1}\|^2 \ ,$$

weil (a) auch für $k+1$ gilt.

(c) Weil (a) auch für $k+1$ gilt, ist für $i < k+1$ mit $\beta_{-1} = 0$

$$(r^i, r^{k+1}) = (-d^i + \beta_{i-1} d^{i-1}, r^{k+1}) = 0 \ .$$

(d) Aus (b) für $k+1$ folgt $d^{k+1} \neq 0$. Für $i < k$ ist

$$\begin{aligned} (d^i, Ad^{k+1}) &= (d^i, A(-r^{k+1} + \beta_k d^k)) \\ &= -(Ad^i, r^{k+1}) + \beta_k (d^i, Ad^k) \\ &= \tfrac{1}{\alpha_i}(r^i - r^{i+1}, r^{k+1}) + \beta_k (d^i, Ad^k) \ . \end{aligned}$$

Dies verschwindet wegen (c) für $k+1$ und (d) für k. Für $i = k$ ist

$$\begin{aligned} (d^k, Ad^{k+1}) &= (Ad^k, d^{k+1}) = \tfrac{1}{\alpha_k}(r^{k+1} - r^k, -r^{k+1} + \beta_k d^k) \\ &= \tfrac{1}{\alpha_k}((r^{k+1}, r^{k+1}) - \beta_k (r^k, d^k)) \end{aligned}$$

wegen (a), (c) für $k+1$, und dies verschwindet nach Definition von β_k.

(e) Es ist sp $\{d^0, \ldots, d^k\}$ = sp $\{r^0, \ldots, r^k\}$ und $d^{k+1} = -r^{k+1} + \beta_k d^k$. Also ist auch sp $\{d^0, \ldots, d^{k+1}\}$ = sp $\{r^0, \ldots, r^{k+1}\}$.

□

Bemerkung: Man nennt Vektoren x, y mit $(x, Ay) = 0$ konjugiert (bezüglich A). Nach Teil (d), (e) des Satzes entstehen die Richtungen d^k aus den Gradienten r^k durch Orthogonalisierung bezüglich des inneren Produktes (x, Ay). Dies erklärt den Namen "konjugierte Gradienten".

Der folgende Satz gibt eine Erklärung für die Effizienz des CG-Verfahrens.

Satz 4.3.3 *Solange das CG-Verfahren nicht abbricht, minimiert x^k die Funktion $f(x)$ in x^0+ sp $\{d^0, \ldots, d^{k-1}\}$.*

Beweis: Sei $f(x)$ minimal für $x = x^0 + \sum_{i=0}^{k-1} c_i d^i$, also

$$\frac{\partial}{\partial c_i} f\left(x^0 + \sum_{i=0}^{k-1} c_i d^i\right) = 0 \ , \qquad i = 0, \ldots, k-1 \ .$$

Wegen

$$\begin{aligned} f\left(x^0 + \sum_{i=0}^{k-1} c_i d^i\right) &= \frac{1}{2}\left(x^0 + \sum_{i=0}^{k-1} c_i d^i, A\left(x^0 + \sum_{i=0}^{k-1} c_i d^i\right)\right) - \left(x^0 + \sum_{i=0}^{k-1} c_i d^i, b\right) \\ &= f(x^0) + \frac{1}{2}\sum_{i,j=0}^{k-1} c_i c_j (d^i, Ad^j) + \sum_{i=0}^{k-1} c_i (d^i, Ax^0 - b) \\ &= f(x^0) + \frac{1}{2}\sum_{i=0}^{k-1} c_i^2 (d^i, Ad^i) + \sum_{i=0}^{k-1} c_i (d^i, r^0) \ , \end{aligned}$$

wo wir Satz 3.2 (d) benutzt haben, gilt

$$c_i = -\frac{(d^i, r^0)}{(d^i, Ad^i)} \ , \qquad i = 0, \ldots, k-1 \ .$$

Nach Konstruktion des CG-Verfahrens ist

$$\begin{aligned} d_i &= -r^i + \beta_{i-1} d^{i-1} \\ &= -r^i - \beta_{i-1}(r^{i-1} + \beta_{i-2} d^{i-2}) \\ &= \cdots \\ &= -r^i - \beta_{i-1} r^{i-1} - \beta_{i-1}\beta_{i-2} r^{i-2} - \cdots + \beta_{i-1} \cdots \beta_0 r^0 \ . \end{aligned}$$

Nach Satz 3.1 (c) gilt also

$$(d^i, r^0) = -\beta_{i-1} \cdots \beta_0 (r^0, r^0) = (r^i, r^i)$$

aufgrund der Definition der β_k. Also haben wir

$$c_i = \frac{(r^i, r^i)}{(d^i, Ad^i)} = \alpha_i \ , \qquad i = 0, \ldots, k-1 \ .$$

Also gilt für den minimalen Vektor x in der Tat

$$x = x^0 + \sum_{i=0}^{k-1} c_i d^i = x^0 + \sum_{i=0}^{k-1} \alpha_i d^i = x^k \ .$$

□

Satz 4.3.4 *Für das CG-Verfahren gilt die Fehlerabschätzung*

$$\|x^k - \overline{x}\|_A \le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|x^0 - \overline{x}\|_A \, ,$$

wo κ *die Kondition von* A *in der euklidischen Norm und* $\|\cdot\|_A$ *die Energie-Norm* $\|x\|_A = (x, Ax)^{1/2}$ *bedeutet.*

Beweis: Nach Satz 3.1 ist $f(x) = \frac{1}{2}\|x - \overline{x}\|_A^2 - \frac{1}{2}\|\overline{x}\|_A^2$. Nach Satz 3.3 minimiert x^k die Funktion $f(x)$ in x^0+ sp $\{d^0, \ldots, d^{k-1}\}$, also minimiert x^k auch $\|x - \overline{x}\|_A$ in diesem Raum, der nach Satz 3.2 identisch ist mit x^0+ sp $\{r^0, \ldots, r^{k-1}\}$. Wir zeigen

$$\mathrm{sp}\{r^0, \ldots, r^{k-1}\} = \mathrm{sp}\{r^0, Ar^0, \ldots, A^{k-1}r^0\} \, . \tag{3.1}$$

Für $k = 1$ ist dies richtig. Sei es richtig für ein $k \ge 1$. Es ist $r^k = r^{k-1} + \alpha_{k-1}Ad^{k-1}$, und nach Satz IV.3.2 (e) ist $d^{k-1} \in$ sp $\{r^0, \ldots, r^{k-1}\} =$ sp $\{r^0, \ldots, A^{k-1}r^0\}$, also $Ad^{k-1} \in$ sp $\{r^0, \ldots, A^k r^0\}$ und damit $r^k \in$ sp $\{r^0, \ldots, A^k r^0\}$. Umgekehrt ist dann auch $A^k r^0 \in$ sp $\{r^0, \ldots, r^k\}$. Damit ist (3.1) gezeigt.

x^k hat also die Gestalt

$$x^k = x^0 + c_0 r^0 + \cdots c_{k-1}A^{k-1}r^0$$

oder

$$\begin{aligned} x^k - \overline{x} &= x^0 - \overline{x} + (c_0 I + \cdots + c_{k-1}A^k)(x^0 - \overline{x}) \\ &= Q(A)(x^0 - \overline{x}) \end{aligned}$$

mit einem Polynom Q der Ordnung k und $Q(0) = 1$. Da A positiv definit ist, existiert $A^{1/2}$, und $A^{1/2}A = AA^{1/2}$. Also ist

$$\begin{aligned} \|x^k - \overline{x}\|_A &= (x^k - \overline{x}, A(x^k - \overline{x}))^{1/2} \\ &= \|A^{1/2}(x^k - \overline{x})\|_2 \\ &= \|A^{1/2}Q(A)(x^0 - \overline{x})\|_2 \\ &= \|Q(A)A^{1/2}(x^0 - \overline{x})\|_2 \\ &\le \|Q(A)\|_2 \|A^{1/2}(x^0 - \overline{x})\|_2 \\ &= \|Q(A)\|_2 \|x^0 - \overline{x}\|_A \, . \end{aligned}$$

Seien $0 < \lambda_1 \leq \cdots \leq \lambda_n$ die Eigenwerte von A. Dann ist

$$\|Q(A)\|_2 = \max_{\lambda_1 \leq \lambda \leq \lambda_n} |Q(\lambda)| \; .$$

Wir suchen nun Q so zu bestimmen, daß $Q(0) = 1$ und $|Q|$ möglichst klein in $[\lambda_1, \lambda_n]$. Dazu führen wir die Tschebitscheff-Polynome T_k erster Art ein. Es ist

$$T_0(x) = 1 \; , \; T_1(x) = x \; , \; T_{k+1}(x) + T_{k-1}(x) = 2xT_k(x) \; .$$

Man bestätigt leicht, daß

$$T_k(x) = \cos kt \quad , \quad x = \cos t \quad , \quad 0 \leq t \leq \pi \; .$$

Wir setzen

$$Q(x) = \frac{T_k(\dfrac{2\lambda - \lambda_1 - \lambda_n}{\lambda_1 - \lambda_n})}{T_k(-\dfrac{\lambda_1 + \lambda_n}{\lambda_1 - \lambda_n})}$$

Offenbar ist Q ein Polynom vom Grade k mit $Q(0) = 1$, und es ist

$$\max_{\lambda_1 \leq \lambda \leq \lambda_n} |Q(\lambda)| = \frac{1}{|T_k(-\dfrac{\lambda_1 - \lambda_n}{\lambda_1 - \lambda_n})|}$$

Nun gilt aber für $x \geq 1$

$$T_k(x) \geq \frac{1}{2}(x + \sqrt{x^2 - 1})^k \; . \tag{3.2}$$

Dazu setzen wir $x = \cosh t$, also $t = \operatorname{arccosh} x = \ell n(x + \sqrt{x^2 - 1})$. Aus der Rekursion für T_k folgt

$$\begin{aligned} T_k(x) &= \cosh kt \\ &\geq \frac{1}{2} e^{kt} \\ &= \frac{1}{2}(x + \sqrt{x^2 - 1})^k \end{aligned}$$

und dies ist (3.2). Mit $x = \frac{\lambda_1 + \lambda_n}{\lambda_1 - \lambda_n} = \frac{1+\kappa}{1-\kappa}$, $\kappa = \kappa(A) = \frac{\lambda_n}{\lambda_1}$ haben wir

$$x + \sqrt{x^2 - 1} = \frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1} \; .$$

Damit folgt die Behauptung.

$\square$

4.4 Vorkonditionierung

Die großen linearen Gleichungssysteme mit dünnbesetzter Matrix, für welche iterative Verfahren besonders geeignet sind, haben häufig eine sehr große Kondition. Wir wollen uns dies an dem Beispiel

$$-\Delta u = f \quad \text{in} \quad \Omega = (0,1)^2 \ , \quad u = 0 \quad \text{auf} \ \partial\Omega$$

klarmachen. Die diskrete Version aus §1 ist

$$\begin{gathered} 4u_{k,\ell} - u_{k+1,\ell} - u_{k-1,\ell} - u_{k,\ell+1} - u_{k,\ell-1} = h^2 f_{k,\ell} \\ k,\ell = 1,\dots,n-1 \ , \quad h = 1/n \ , \\ u_{k,\ell} = 0 \quad \text{falls} \quad k \in \{0,n\} \ \text{ oder } \ \ell \in \{0,n\} \ . \end{gathered} \tag{4.1}$$

Dabei bedeutet $u_{k,\ell}$ eine Näherung für u in dem Gitterpunkt $(hk, h\ell)$. Wir numerieren die Gitterpunkte und die Gleichungen (4.1) etwa zeilenweise von 1 bis $N = (n-1)^2$. Man sieht dann leicht, daß die Matrix A von (4.1) eine dünnbesetzte symmetrische (N,N)-Matrix ist.

Wir berechnen die Eigenwerte von A. Mit $v_k = \sin \alpha\pi kh$ ist wegen der Additionstheoreme

$$2v_k - v_{k+1} - v_{k-1} = 2(1 - \cos \alpha\pi h) v_k \ .$$

Mit $v_{k,\ell} = \sin \alpha\pi kh \sin \beta\pi \ell h$ ist also

$$4v_{k,\ell} - v_{k+1,\ell} - v_{k-1,\ell} - v_{k,\ell+1} - v_{k,\ell-1} = 2(2 - \cos \alpha\pi h - \cos \beta\pi h) v_{k,\ell} \ .$$

Wählen wir nun α, β ganz und $1 \le \alpha$, $\beta < n$, so ist $v_{k,\ell} = 0$ für $k \in \{0,n\}$ oder $\ell \in \{0,n\}$ und damit der Vektor $(v_{k,\ell})_{k,\ell=1,\dots,n-1}$ Eigenvektor von A zum Eigenwert

$$\lambda_{\alpha,\beta} = 2(2 - \cos \alpha\pi h - \cos \beta\pi h) \ .$$

Damit haben wir alle N Eigenwerte von A gefunden. Die Kondition $\kappa(A)$ von A bezüglich der euklidischen Norm berechnet sich nun als Quotient von größtem und kleinstem Eigenwert von A, also

$$\kappa(A) = \frac{\lambda_{n-1,n-1}}{\lambda_{1,1}} = \frac{1 + \cos h\pi}{1 - \cos h\pi} \ .$$

Für kleine h verhält sich dies wie $4/(h\pi)^2$, ist also sehr groß.

Berechnen wir die Konvergenzgeschwindigkeit des Gesamtschrittverfahrens! Die Iterationsmatrix ist $B = -D^{-1}(L+R)$ mit $A = D+L+R$, $D = 4I$. Sei μ ein Eigenwert von B. Dann ist $4(1-\mu)$ Eigenwert von A, also $\mu = 1-\lambda/4$. Die Eigenwerte von B sind also

$$\mu_{\alpha,\beta} = 1 - \lambda_{\alpha,\beta}/4 = \frac{1}{2}(\cos\alpha\pi h + \cos\beta\pi h) \ .$$

Damit erhalten wir

$$\rho(B) = \max_{1\le\alpha,\beta<n} |\mu_{\alpha,\beta}| = \cos\pi h \ .$$

Für kleine h verhält sich dies wie $1-\pi^2h^2/2$. Die Konvergenz des Gesamtschrittverfahrens ist also sehr langsam.

Gesamtschritt- und CG-Verfahren konvergieren also beide langsam, doch mit einem bedeutenden Unterschied: Während sich der Fehler beim Gesamtschrittverfahren wie $(1-\pi^2h^2/2)^k$ verhält, verhält er sich beim CG-Verfahren (allerdings in einer anderen Norm) wie $(1-h\pi)^k$. Das CG-Verfahren konvergiert also weniger langsam als das Gesamtschrittverfahren. Man kann zeigen, daß das SOR-Verfahren bei optimaler Wahl von ω ungefähr so schnell wie das CG-Verfahren konvergiert. Insgesamt muß man also bei allen diesen Verfahren mit langsamer Konvergenz rechnen, und zwar wird die Konvergenz umso langsamer, je kleiner h wird.

Abhilfe schafft hier die Vorkonditionierung. Dazu schreiben wir $Ax = b$ in der Form

$$C^{-1}A(C^*)^{-1}C^*x = C^{-1}b \tag{4.2}$$

mit einer nichtsingulären Matrix C. C wird so gewählt, daß

$$\kappa(C^{-1}A(C^*)^{-1}) \ll \kappa(A) \ .$$

(4.2) hat dann die Gestalt

$$\tilde{A}\tilde{x} = \tilde{b} \ ,$$

$$\tilde{A} = C^{-1}A(C^*)^{-1} \ , \quad \tilde{x} = C^*x \ , \quad \tilde{b} = C^{-1}b \ ,$$

wobei $\tilde{A}$ eine viel bessere Kondition hat als A. Überdies ist $\tilde{A}$ hermitesch (positiv definit), falls dies für A der Fall ist.

Wegen

$$\begin{aligned}
(C^{-1})^*\tilde{A}C^* &= (C^{-1})^* C^{-1} A (C^*)^{-1} C^* = M^{-1}A\,,\\
M &= CC^*
\end{aligned}$$

ist $\kappa(\tilde{A}) = \kappa(M^{-1}A)$. Um für $\tilde{A}$ eine gute Kondition zu bekommen, wird man also $M \sim A$ wählen wollen. Gleichzeitig muß M aber eine Cholesky-Faktorisierung $M = CC^*$ besitzen, so daß C^{-1}, $(C^*)^{-1}$ leicht auf Vektoren anwendbar ist. Für $A = I + L + L^*$ ist

$$M = (I + L)(I + L^*)$$

eine Möglichkeit.

Andere Möglichkeiten sind Dekompositionsmethoden und Mehrgitterverfahren. Diese führen zu von h unabhängigen Konvergenzraten. Sie sind Gegenstand weiterführender Vorlesungen, wie etwa "Numerik partieller Differentialgleichungen".

Kapitel 5

Interpolation

5.1 Interpolation durch Polynome

Sei $\mathcal{P}_n$die Menge der Polynome mit komplexen Koeffizienten vom Grade $\leq n$. $\mathcal{P}_n$ besteht also aus den Ausdrücken der Form

$$\sum_{k=0}^{n} a_n x^k \quad , \quad a_n \in \mathbb{C} \ .$$

Als Polynominterpolation bezeichnet man folgende Aufgabe: Gegeben seien $n+1$ "Stützstellen" $x_0, \ldots, x_n \in \mathbb{C}$ und ebensoviel "Stützwerte" $y_0, \ldots, y_n \in \mathbb{C}$. Gesucht ist $p \in \mathcal{P}_n$ mit $p(x_j) = y_j$, $j = 0, \ldots, n$.

Satz 5.1.1 *p ist eindeutig bestimmt, falls die x_j paarweise verschieden sind.*

Beweis: Das Interpolationsproblems ist äquivalent dem linearen Gleichungssystem

$$\sum_{k=0}^{n} a_k x_j^k = y_j \quad , \quad j = 0, \ldots, n \tag{1.1}$$

für $a_0, \ldots, a_n$. Wir zeigen, daß dieses eindeutig lösbar ist. Dazu ist hinreichend, daß das zugehörige homogene System, also das System mit $y_0 = y_1 = \cdots = y_n = 0$, nur die Lösung $a_0 = a_1 = \cdots a_n = 0$ hat. Dies ist aber der Fall, weil ein Polynom vom Grade $\leq n$ nicht mehr als $n+1$ paarweise verschiedene Nullstellen haben kann.

□

Zur Berechnung des interpolierenden Polynoms könnte man das lineare Gleichungssystem (1.1) lösen, etwa durch die Cramer'sche Regel. Die Determinante von (1.1) ist die Vandermond'sche Determinante

$$V = \begin{vmatrix} 1 & \cdots & 1 \\ x_0 & & x_n \\ \vdots & & \vdots \\ x_0^n & & x_n^n \end{vmatrix} = \prod_{i>k}(x_i - x_k) \; .$$

Diese ist also $\neq 0$, falls die x_j paarweise verschieden sind, was einen neuen Beweis von Satz 1 liefert. Wir ersetzen in V die k-te Spalte durch die Zahlen $y_0, \ldots, y_n$ und bezeichnen das Resultat mit V_k. Die Lösung des Interpolationsproblems ist dann

$$p(x) = \sum_{k=0}^{n} a_k x^k \quad , \quad a_k = \frac{V_k}{V} \, , \quad k = 0, \ldots, n \; .$$

Für das praktische Rechnen ist diese Lösung völlig ungeeignet. Wir werden drei sehr viel praktischere Darstellungen von p finden.

1. Die Form von Lagrange

Man setzt

$$\omega_j(x) = \prod_{i=0,\, i\neq j}^{n} \frac{x - x_i}{x_j - x_i} \quad , \quad j = 0, \ldots, n \; .$$

Es ist $\omega_j \in \mathcal{P}_n$ Lösung des speziellen Interpolationsproblems

$$\omega_j(x_k) = \begin{cases} 1 & , \quad k = j \; , \\ 0 & , \quad \text{sonst} \; . \end{cases}$$

Für das gesuchte Polynom gilt dann

$$p(x) = \sum_{j=0}^{n} y_j \omega_j(x) \; .$$

Beispiel: $n = 2$. Stützstellen und Stützwerte seien

j	x_j	y_j
0	0	1
1	1	3
2	3	2

Wir berechnen

$$\begin{aligned}
\omega_0(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} &= \tfrac{1}{3}(x-1)(x-3)\\
\omega_1(x) &= \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} &= -\tfrac{1}{2}x(x-3)\\
\omega_2(x) &= \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} &= \tfrac{1}{6}x(x-1)
\end{aligned}$$

und erhalten

$$p(x) = \frac{1}{3}(x-1)(x-3) - \frac{3}{2}x(x-3) + \frac{1}{3}x(x-1) = -\frac{5}{6}x^2 + \frac{17}{6}x + 1 \ .$$

2. Die Rekursionsformel von Neville

Wir bezeichnen mit $p_{i,\ldots,k}$ das nach Satz 1 eindeutig bestimmte Polynom $\in \mathcal{P}_{k-i}$, welches $y_i, \ldots, y_k$ an den Stellen $x_i, \ldots, x_k$ interpoliert. Das gesuchte Polynom ist dann $p = p_{0,\ldots,n}$.

Satz 5.1.2 *Es ist*

$$p_{i,\ldots,k}(x) = \frac{1}{x_k - x_i}\ \left((x-x_i)p_{i+1,\ldots,k}(x) + (x_k - x)p_{i,\ldots,k-1}(x)\right) \ .$$

Beweis: Auf der rechten Seite steht ein Polynom vom Grade $k-i$, das an den Stellen $x_i, \ldots, x_k$ die Werte $y_i, \ldots, y_k$ annimmt. Nach Satz 1 ist dieses eindeutig bestimmt und muß daher mit $p_{i,\ldots,k}$ übereinstimmen.

□

Die Neville'sche Formel erlaubt die Berechnung des Interpolationspolynoms p nach folgendem Schema ($n = 3$):

$$\begin{array}{ccccccc}
x_0 & y_0 & = & p_0 & & & \\
 & & & & p_{0,1} & & \\
x_1 & y_1 & = & p_1 & & p_{0,1,2} & \\
 & & & & p_{1,2} & & p_{0,1,2,3} = p \\
x_2 & y_2 & = & p_2 & & p_{1,2,3} & \\
 & & & & p_{2,3} & & \\
x_3 & y_3 & = & p_3 & & &
\end{array}$$

Bei Hinzunahme einer weiteren Stützstelle (und Erhöhung des Polynomgrads) wird das Schema einfach um eine Zeile erweitert, ohne daß die bereits berechneten Teile des Schemas neu berechnet werden müßten.

3. Die Newton'sche Form

Für $p = p_{0,\ldots,n}$ machen wir den Ansatz

$$\begin{aligned}
p_{0,\ldots,n}(x) \quad = \quad & A_0 + A_1(x - x_0) + A_2(x - x_0)(x - x_1) + \cdots \\
& + A_n(x - x_0)(x - x_1) \cdots (x - x_{n-1})
\end{aligned}$$

mit noch zu bestimmenden Koeffizienten $A_0, \ldots, A_n$. Die Bedingungen $p(x_j) = y_j,\ \ j = 0, \ldots, n$ ergeben für die A_k das lineare Gleichungssystem

$$\begin{aligned}
A_0 \quad & = \quad y_0 \, , \\
A_0 + A_1(x_1 - x_0) \quad & = \quad y_1 \, , \\
A_0 + A_1(x_2 - x_0) + A_2(x_2 - x_0)(x_2 - x_1) \quad & = \quad y_2 \, , \\
\cdots \quad & \\
A_0 + A_1(x_n - x_0) + \cdots + A_n(x_n - x_0) \cdots (x_n - x_{n-1}) \quad & = \quad y_n \, .
\end{aligned}$$

Die A_k können also durch Vorwärtseinsetzen berechnet werden. Für das Resultat gibt es eine elegante Darstellung durch "dividierte Differenzen" $[y_i, \ldots, y_k]$. Diese sind rekursiv definiert durch

$$\begin{aligned}
[y_i] \quad & = \quad y_i \, , \\
[y_i, \ldots, y_k] \quad & = \quad \frac{1}{x_k - x_i}([y_{i+1}, \ldots, y_k] - [y_i, \ldots, y_{k-1}]) \, .
\end{aligned}$$

Z.B. ist

$$\begin{aligned}
[y_0, y_1] &= \frac{y_1 - y_0}{x_1 - x_0} \,, \\
[y_0, y_1, y_2] &= \frac{1}{x_2 - x_0} \left(\frac{y_1 - y_2}{x_1 - x_2} - \frac{y_0 - y_1}{x_0 - x_1} \right) \,.
\end{aligned}$$

Man ordnet die dividierten Differenzen im “Differenzenschema” an:

$$\begin{array}{lllll}
x_0 & [y_0] & & & \\
 & & [y_0, y_1] & & \\
x_1 & [y_1] & & [y_0, y_1, y_2] & \\
 & & [y_1, y_2] & & [y_0, y_1, y_2, y_3] \\
x_2 & [y_2] & & [y_1, y_2, y_3] & \\
 & & [y_2, y_3] & & \\
x_3 & [y_3] & & &
\end{array}$$

Satz 5.1.3 *Für die Koeffizienten A_i gilt*

$$A_i = [y_0, \ldots, y_i] \,, \qquad i = 0, \ldots, n \,.$$

Beweis: Wir zeigen

$$\begin{aligned}
p_{i,\ldots,k}(x) &= [y_i] + [y_i, y_{i+1}](x - x_i) + [y_i, y_{i+1}, y_{i+2}](x - x_i)(x - x_{i+1}) \\
&\quad + \cdots + [y_i, \ldots, y_k](x - x_i) \cdots (x - x_{k-1})
\end{aligned} \tag{1.2}$$

durch Induktion nach $m = k - i$. Für $m = 0$ ist (1.2) richtig. Sei (1.2) richtig für ein $m \geq 0$. Dann ist

$$\begin{aligned}
p_{i,\ldots,k}(x) &= [y_i, \ldots, y_k](x - x_i) \cdots (x - x_{k-1}) + q_1(x) \\
&= [y_i, \ldots, y_k] x^{k-i} + q_2(x)
\end{aligned}$$

mit $q_\ell \in \mathcal{P}_{k-i-1}$, und ebenso gilt

$$p_{i+1,\ldots,k+1}(x) = [y_{i+1}, \ldots, y_{k+1}] x^{k-i} + q_3(x) \,.$$

Nach Satz 2 ist mit $r_\ell \in \mathcal{P}_{k-i}$

$$\begin{aligned}
p_{i,\dots,k+1}(x) &= \frac{1}{x_{k+1}-x_i}((x-x_i)p_{i+1,\dots,k+1}(x) + (x_{k+1}-x)p_{i,\dots,k}(x)) \\
&= \frac{1}{x_{k+1}-x_i}([y_{i+1},\dots,y_{k+1}] - [y_i,\dots,y_k])x^{k-i+1} + r_1(x) \\
&= [y_i,\dots,y_{k+1}]x^{k-i+1} + r_1(x) \\
&= [y_i,\dots,y_{k+1}](x-x_i)\cdots(x-x_k) + r_2(x)\ .
\end{aligned}$$

Offenbar muß $r_2(x_j) = y_j,\ \ j = i,\dots,k$ sein. Nach Satz 1 ist also $r_2 = p_{i,\dots,k}$. Da für $p_{i,\dots,k}$ (1.2) bereits gilt, gilt (1.2) auch für $p_{i,\dots,k+1}$.

□

Beispiel: Wir haben oben das Interpolationsproblem

j	x_j	y_j
0	0	1
1	1	3
2	3	2

durch die Lagrange'sche Form des Interpolationspolynoms gelöst. Zur Lösung des gleichen Problems mit der Newton'schen Form stellen wir zunächst einmal das Differenzenschema auf:

$$\begin{array}{cccc}
0 & 1 & & \\
 & & 2 & \\
1 & 3 & & -5/6 \\
 & & -1/2 & \\
3 & 2 & &
\end{array}$$

Die Koeffizienten des Newton'schen Interpolationspolynoms stehen in der obersten Zeile:

$$p(x) = 1 + 2x - \frac{5}{6}x(x-1) = 1 + \frac{17}{x}x - \frac{5}{6}x^2\ .$$

Fügen wir noch die Stützstelle $x_3 = 3$ mit dem Stützwert $y_3 = 4$ hinzu, so lautet das erweiterte Differenzenschema

$$\begin{array}{ccccc}
0 & 1 & & & \\
 & & 2 & & \\
1 & 3 & & -5/6 & \\
 & & -1/2 & & -1/3 \\
3 & 2 & & -3/2 & \\
 & & -2 & & \\
2 & 4 & & &
\end{array}$$

und das zugehörige Interpolationsproblem ist

$$p(x) = 1 + 2x - \frac{5}{6}x(x-1) - \frac{1}{3}x(x-1)(x-3) \; .$$

5.2 Der Interpolationsfehler

Seien in einem Intervall $[a, b]$ $n + 1$ paarweise verschiedene Stützstellen $x_0, \ldots, x_n$ und eine Funktion $f \in C_{n+1}[a, b]$ gegeben. Wir wollen $f(x)$ für $x \neq x_i$ approximieren.

Betrachten wir das eindeutig bestimmte Polynom $p \in \mathcal{P}_n$ mit $p(x_j) = f(x_j)$, $j = 0, \ldots, n$. Der folgende Satz macht eine Aussage über den Fehler, der bei einer Approximation von f durch p auftritt:

Satz 5.2.1 *Zu jedem $x \in [a, b]$ existiert ein $\tilde{x} \in [a, b]$, so daß gilt:*

$$f(x) - p(x) = w(x) \frac{f^{(n+1)}(\tilde{x})}{(n+1)!}$$

mit

$$w(x) = \prod_{j=0}^{n} (x - x_j)$$

Beweis: Sei $\overline{x} \in [a, b]$ beliebig gewählt mit $\overline{x} \neq x_j$, $j = 0, \ldots, n$. Wir setzen

$$F(x) = f(x) - p(x) - Kw(x)$$

mit einer Konstanten K. Dann ist

$$F(x_j) = 0 \quad , \quad j = 0, \ldots, n \; .$$

Wir wählen K nun so, daß auch $F(\overline{x})$ verschwindet, also

$$K = \left(\frac{f - p}{w} \right) (\overline{x}) \; .$$

Damit hat F in $[a, b]$ mindestens die $n + 2$ Nullstellen $\overline{x}$, $x_0, \ldots, x_n$. Durch wiederholte Anwendung des Satzes von Rolle folgt, daß $F^{(n+1)}$ mindestens eine Nullstelle $\tilde{x}$ in $[a, b]$ besitzt.

Aus der Beziehung

$$F^{(n+1)}(x) = f^{(n+1)}(x) - K(n+1)!$$

folgt

$$K = \left(\frac{f-p}{w}\right)(\overline{x}) = \frac{f^{(n+1)}(\tilde{x})}{(n+1)!} \; .$$

Also

$$f(\overline{x}) - p(\overline{x}) = w(\overline{x}) \, \frac{f^{(n+1)}(\tilde{x})}{(n+1)!} \; .$$

Diese Beziehung ist offenbar auch für $\overline{x} = x_j, \;\; j = 0, \ldots, n$ erfüllt.

□

Für den Interpolationsfehler erhalten wir die Abschätzung

$$|f(x) - p(x)| \leq |w(x)| \, \max_{x \in [a,b]} \frac{|f^{(n+1)}(x)|}{(n+1)!} \; .$$

1) Wir betrachten zuerst den Fall gleichmäßig verteilter Stützstellen $x_j = a + jh$ mit der "Schrittweite" $h = \frac{b-a}{n}$. Für $x = a + \theta h$, $0 \leq \theta \leq n$ gilt

$$w(x) = h^{n+1} \prod_{j=0}^{n} (\theta - j) \; ,$$

also

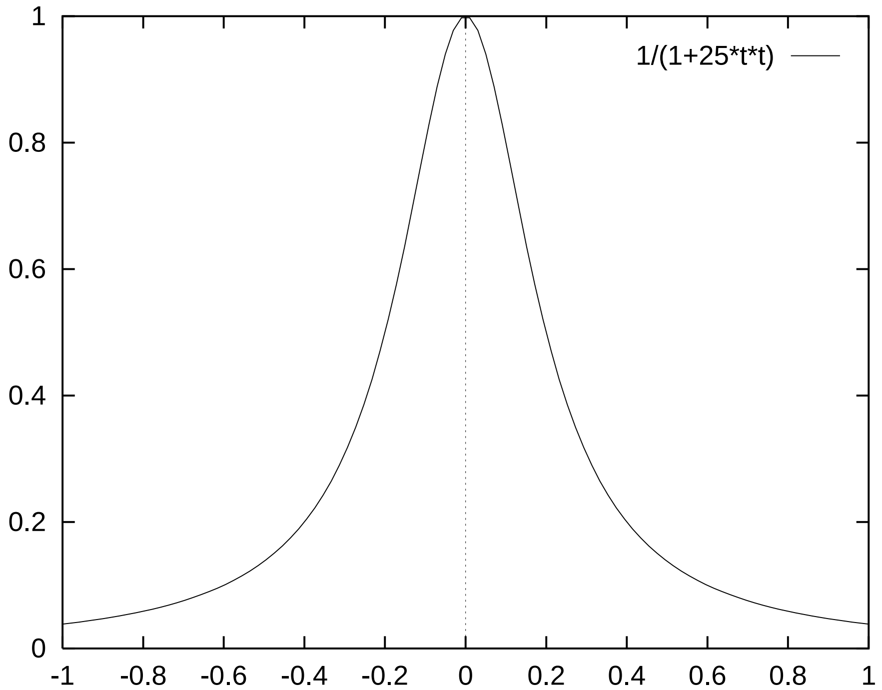
$$|f(x) - p(x)| \leq \frac{h^{n+1}}{(n+1)!} \left(\prod_{j=0}^{n} |\theta - j| \right) \max_{x \in [a,b]} |f^{(n+1)}(x)| \; .$$

(a) Für $b - a \to 0$ bei festem n erhalten wir

$$f(x) - p(x) = \mathrm{O}(h^{n+1}) \; .$$

(b) Der Fall $n \to \infty$ bei fester Intervallänge $b - a$ führt i.a. zu keiner Konvergenz. Wir betrachten hierzu das Beispiel von Runge:

$$f(x) = (1 + 25x^2)^{-1} \quad \text{in} \;\; [-1, 1]$$



f wird an den Stellen $x_j = -1 + \frac{2j}{n}$, $j = 0, \ldots, n$ durch ein Polynom vom Grad n interpoliert. Die folgende Tabelle zeigt, daß der Interpolationsfehler für große n stark anwächst.

n	$\max_{x\in[-1,1]} \lvert f(x) - p(x) \rvert$
1	$0,96$
5	$0,43$
13	$1,07$
19	$8,57$

2) Wir wählen die Stützstellen $x_0, \ldots, x_n$ nun so, daß $\max\limits_{[a,b]} |w(x)|$ möglichst klein ist.

Für $[a, b] = [-1, 1]$ erhalten wir

$$w(x) = 2^{-n} T_{n+1}(x) \; ,$$

wobei für $x \in [-1, 1]$ die Tschebyscheff-Polynome T_n wie folgt definiert sind:

$$T_n(x) = \cos nt \quad , \quad x = \cos t \quad , \quad 0 \leq t \leq \pi \; .$$

In IV.3 haben wir gesehen, daß

$$T_{n+1} = 2xT_n - T_{n-1} \ .$$

Die Rekursion zeigt, daß T_n für $n \geq 1$ die Form

$$T_n(x) = 2^{n-1}x^n + \text{Polynom} \quad \in \mathcal{P}_{n-1}$$

haben muß.
Daher hat $w(x) = 2^{-n}T_{n+1}(x)$ den Höchstkoeffizienten 1 und es gilt

$$|w(x)| \leq 2^{-n} \quad \text{in } \ [-1,1] \ .$$

Die Nullstellen $x_j = \cos\frac{(j+1/2)\pi}{n}$, $j = 0, \ldots, n$ von w sind unsere Stützstellen.
Bei dieser Wahl ergibt sich die Fehlerabschätzung

$$|f(x) - p(x)| \leq \frac{\text{Max}|f^{(n+1)}(x)|}{2^n(n+1)!} \ .$$

Für das obige Beispiel ergeben sich folgende Werte:

n	$\max_{x\in[-1,1]} \lvert f(x) - p(x)\rvert$
1	0,93
5	0,56
13	0,12
19	0,04

Die Verbesserung ist erheblich, die Approximation aber immer noch unbefriedigend.

5.3 Trigonometrische Interpolation

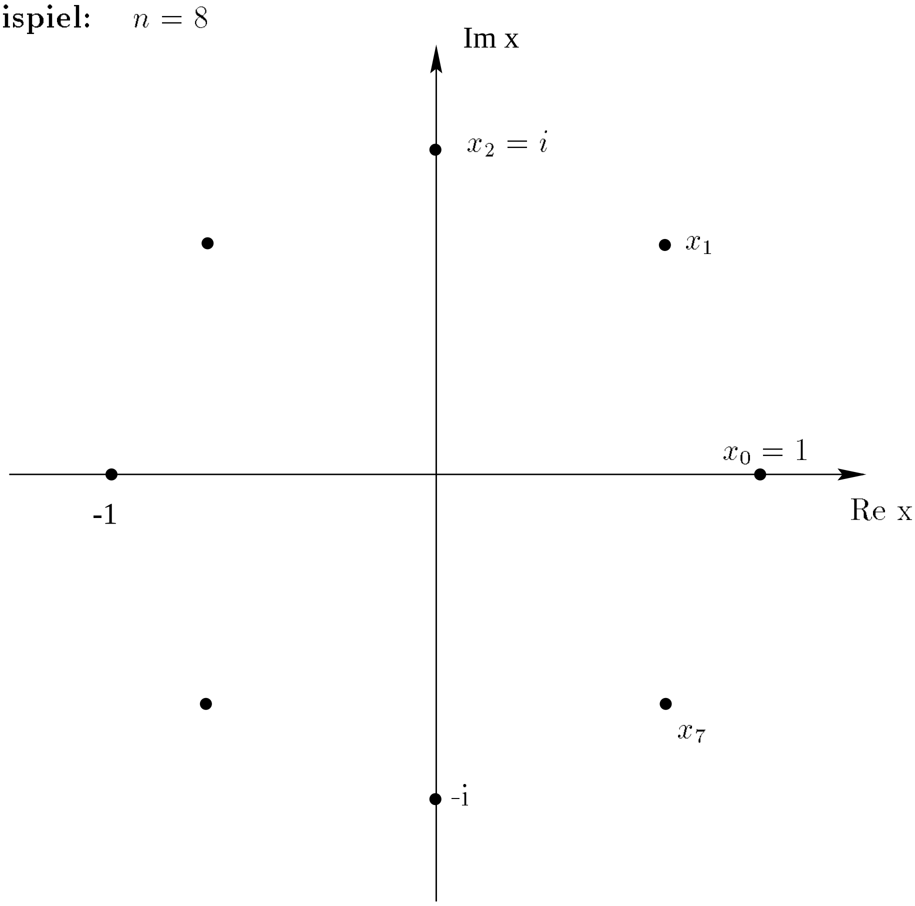
Wir betrachten in diesem Abschnitt einen wichtigen Spezialfall der Polynominterpolation, bei dem die Stützstellen in regelmäßigen Abständen auf dem komplexen Einheitskreis liegen.

Diese spezielle Problemstellung wird zu einem der wichtigsten Hilfsmittel der angewandten Mathematik führen: Zur diskreten Fouriertransformation.

Die Stützstellen seien gegeben durch

$$x_j = e^{it_j} = \cos t_j + i \sin t_j \; ; \quad t_j = 2\pi j/n \; ; \quad j = 0, \ldots, n-1 \; .$$

Beispiel: $n = 8$



Nach Satz 5.1.1 gibt es ein eindeutig bestimmtes Polynom $p \in \mathcal{P}_{n-1}$ mit $p(x_j) = y_j,\ \ j = 0, \ldots, n-1$.

Die Koeffizienten von p bezeichnen wir mit $\hat{y}_k$:

$$p(x) = \sum_{k=0}^{n-1} \hat{y}_k x^k$$

Seien $y = (y_0, \ldots, y_{n-1})^T \in \mathbb{C}^n$ und $\hat{y} = (\hat{y}_0, \ldots, \hat{y}_{n-1})^T \in \mathbb{C}^n$. Dann können wir die Interpolationsaufgabe

$$y_j = \sum_{k=0}^{n-1} \hat{y}_k x_j^k \ ; \quad j = 0, \ldots, n-1$$

in die Form

$$y = \begin{pmatrix} 1 & x_0 & \ldots & x_0^{n-1} \\ 1 & x_1 & \ldots & x_1^{n-1} \\ \vdots & & \ddots & \\ 1 & x_{n-1} & \ldots & x_{n-1}^{n-1} \end{pmatrix} \hat{y} = W\hat{y} \ .$$

umschreiben. Die Inversion der Matrix W erweist sich als sehr einfach:

Satz 5.3.1 *Mit obigen Definitionen für x_j, $j = 0, \ldots, n-1$ gilt für die Matrix W:*

$$WW^* = nI \ .$$

Beweis: Für k, $\ell = 0, \ldots, n-1$ gilt:

$$\begin{aligned} (WW^*)_{k\ell} &= \sum_{j=0}^{n-1} x_k^j \overline{x}_\ell^j \\ &= \sum_{j=0}^{n-1} e^{i(t_k - t_\ell)j} \\ &= \sum_{j=0}^{n-1} e^{2\pi i(k-\ell)j/n} \\ &= \sum_{j=0}^{n-1} q^j \qquad \text{mit } \ q = e^{2\pi i(k-\ell)/n} \end{aligned}$$

$$= \begin{cases} \frac{q^n-1}{q-1} & , \text{ falls } q \neq 1 \\ n & , \text{ falls } q = 1 \end{cases}$$

$$= \begin{cases} 0 & , \text{ falls } k \neq \ell \\ n & , \text{ falls } k = \ell \end{cases}$$

□

Damit haben wir gleichzeitig eine Orthogonalitätseigenschaft der trigonometrischen Funktionen gezeigt:

$$\frac{1}{n}\sum_{j=0}^{n-1} e^{2\pi ijk/n} = \begin{cases} 1 & , \text{ falls } k \in n\mathbb{Z} \\ 0 & , \text{ sonst} \end{cases}$$

Aus Satz 5.3.1 können wir folgern:

$$W^{-1} = \frac{1}{n} W^*$$

und damit

$$\hat{y} = \frac{1}{n} W^* y \ .$$

In Komponenten:

$$\boxed{\begin{array}{rcll} \hat{y}_k & = & \frac{1}{n}\sum\limits_{j=0}^{n-1} e^{-2\pi ijk/n} y_j \ , & (1) \\ y_j & = & \sum\limits_{k=0}^{n-1} e^{2\pi ijk/n} \hat{y}_k \quad , & (2) \end{array}}$$

Gleichung (1) heißt diskrete Fouriertransformation der Länge n, (2) heißt

entsprechend inverse diskrete Fouriertransformation der Länge n. Beide werden in der Angewandten Mathematik sehr häufig benutzt. Man programmiert jedoch nicht nach den Formeln (1) und (2), was jeweils n^2 komplexe Rechenoperationen beanspruchen würde, sondern benutzt erheblich schnellere Algorithmen. Einen davon werden wir in 5.4 kennenlernen.

Zunächst stellen wir jedoch die Beziehung zum Titel dieses Paragraphen her und drücken $p(x)$ durch trigonometrische Funktionen aus:

Wir setzen

$$\begin{aligned} a_k &= \frac{2}{n} \sum_{j=0}^{n-1} y_j \cos(2\pi kj/n) , \\ b_k &= \frac{2}{n} \sum_{j=0}^{n-1} y_j \sin(2\pi kj/n) . \end{aligned}$$

Dann ist

$$\begin{aligned} \hat{y}_k &= \frac{1}{n} \sum_{j=0}^{n-1} y_j e^{-2\pi ijk/n} \\ &= \frac{1}{n} \sum_{j=0}^{n-1} y_j (\cos(2\pi jk/n) - i \sin(2\pi jk/n)) \\ &= \frac{1}{2}(a_k - ib_k) . \end{aligned}$$

Man beachte, daß dies keine Zerlegung in Real- und Imaginärteil von $\hat{y}_k$ darstellt, da die a_k und b_k nicht notwendig reell sind. Es gilt:

$$a_{n-k} = a_k \quad \text{und} \quad b_{n-k} = -b_k$$

und damit

$$\hat{y}_{n-k} = \frac{1}{2}(a_k + ib_k) .$$

Sei n nun ungerade, also $n = 2m + 1$. Dann erhalten wir

$$\begin{aligned} p(x_j) &= \sum_{k=0}^{n-1} \hat{y}_k e^{2\pi ijk/n} \\ &= \hat{y}_0 + \sum_{k=1}^{m} \hat{y}_k e^{2\pi ijk/n} + \sum_{k=m+1}^{n-1} \hat{y}_k e^{2\pi ijk/n} \\ &= \hat{y}_0 + \sum_{k=1}^{m} \hat{y}_k e^{2\pi ijk/n} + \sum_{k'=1}^{m} \hat{y}_{n-k'} e^{2\pi ij(n-k')/n} \quad , \quad k' = n - k \\ &= \hat{y}_0 + \frac{1}{2} \sum_{k=1}^{m} \{(a_k - ib_k)(\cos(2\pi jk/n) + i \sin(2\pi jk/n)) \\ &\qquad + (a_k + ib_k)(\cos(2\pi jk/n) - i \sin(2\pi jk/n))\} , \end{aligned}$$

also
$$p(x_j) = \frac{a_0}{2} + \sum_{k=1}^{m}(a_k \cos(2\pi jk/n) + b_k \sin(2\pi jk/n)) \ .$$
Für gerades n, $n = 2m$, erhält man analog
$$\begin{aligned} p(x_j) = \frac{a_0}{2} \quad &+ \quad \sum_{k=1}^{m-1}(a_k \cos(2\pi jk/n) + b_k \sin(2\pi jk/n)) \\ &+ \quad \frac{a_m}{2} \cos(2\pi jm/n) \ , \end{aligned}$$
wenn man beachtet, daß b_m verschwindet.

Wir bezeichnen die Menge
$$\mathcal{T}_m = \left\{ \frac{a_0}{2} + \sum_{k=1}^{m}(a_k \cos kt + b_k \sin kt) \ ; \ a_k, b_k \in \mathbb{C} \right\}$$
als die Menge der trigonometrischen Polynome vom Grad m.

Damit haben wir die folgende Aussage über die trigonometrische Interpolation gewonnen:

Satz 5.3.2 *Gesucht sei $T \in \mathcal{T}_m$ mit*
$$\begin{aligned} T(t_j) &= y_j \quad , \quad j = 0, \ldots, n-1 \\ t_j &= 2\pi j/n \ . \end{aligned}$$
Für $n = 2m+1$ ist die Aufgabe eindeutig lösbar und die Lösung ist durch
$$T(t) = \frac{a_0}{2} + \sum_{k=1}^{m}(a_k \cos kt + b_k \sin kt)$$
gegeben.

Für $n = 2m$ gibt es genau eine Lösung mit $b_m = 0$, die durch
$$T(t) = \frac{a_0}{2} + \sum_{k=1}^{m-1}(a_k \cos kt + b_k \sin kt) + \frac{a_m}{2} \cos mt$$
gegeben ist. Dabei sind a_k, b_k die oben angegebenen Summen.

5.4 Schnelle Fouriertransformation

Wir betrachten nun einen effizienten Algorithmus zur Berechnung von $\hat{y}$, die schnelle Fouriertransformation (FFT) von Cooley-Tukey.

Sei n gerade, $n = 2m$. Dann gilt

$$\hat{y}_k = \sum_{j=0}^{n-1} y_j q^{jk} \quad \text{mit} \quad q = e^{-2\pi i/n} \ .$$

Es sind $q^n = 1$ und $q^m = q^{n/2} = -1$.

Die Idee ist nun, die Summe nach geraden und ungeraden Indizes zu zerlegen:

$$\begin{aligned}
\hat{y}_k &= \sum_{\ell=0}^{m-1} q^{(2\ell)k} y_{2\ell} + \sum_{\ell=0}^{m-1} q^{(2\ell+1)k} y_{2\ell+1} \\
&= \sum_{\ell=0}^{m-1} (q^2)^{\ell k} y_{2\ell} + q^k \sum_{\ell=0}^{m-1} (q^2)^{\ell k} y_{2\ell+1} \\
&=: g_k + q^k u_k
\end{aligned}$$

Wir sehen nun, daß sich g_k und u_k als Fouriertransformationen der Länge $m = n/2$ berechnen, da gilt:

$$q^2 = e^{-2\pi i/(n/2)}$$

Weiter haben wir

$$\begin{aligned}
g_{k+m} &= g_k \\
u_{k+m} &= u_k
\end{aligned}$$

und damit für $k = 0, \ldots, m-1$

$$\begin{aligned}
\hat{y}_k &= g_k + q^k u_k \ , \\
\hat{y}_{k+m} &= g_k + q^{k+m} u_k = g_k - q^k u_k \ .
\end{aligned}$$

Sei nun M_p die Anzahl der komplexen Multiplikationen und A_p die Anzahl der komplexen Additionen, die für eine schnelle Fouriertransformation der Länge

$n = 2^p$ benötigt werden. Wenn wir die Berechnung von q^k vernachlässigen, erhalten wir $A_0 = 0$, $M_0 = 0$ und

$$\begin{aligned} M_{p+1} &= 2M_p + 2^p \ , \\ A_{p+1} &= 2A_p + 2^{p+1} \ , \end{aligned}$$

also

$$\begin{aligned} M_p &= \tfrac{1}{2}p2^p &= \tfrac{1}{2}(\log_2 n) \cdot n \\ A_p &= p2^p &= (\log_2 n) \cdot n \ . \end{aligned}$$

Damit gilt der

Satz 5.4.1 *Die Fouriertransformation der Länge $n = 2^p$ kann durch $\frac{1}{2}n \log_2 n$ komplexe Multiplikationen und $n \log_2 n$ komplexe Additionen berechnet werden.*

Die Implementierung der FFT durch ein rekursives Programm ist sehr einfach:

```
fft (y, n)
/⋆ Führt die FFT der Länge n = 2^p durch ⋆/
{ m = n/2;
   for ℓ = 0, ..., m − 1 {
        g[ℓ] = y[2ℓ];
        u[ℓ] = y[2ℓ + 1];
   }
   if (m > 1) {
        fft (g, m);
        fft (u, m);
   }
   for k = 0, ..., m − 1 {
        u[k] = q^k ∗ u[k];
        y[k] = g[k] + u[k];
        y[k + m] = g[k] − u[k];
   }
}
```

Die FFT nach Cooley-Tukey ist ein typisches Beispiel für das "divide and conquer"-Prinzip der Informatik:

1. Zerlege das Problem in Teilprobleme.
2. Löse die Teilprobleme.
3. Setze die Lösung der Teilprobleme zur Lösung des ganzen Problems zusammen.

Beispiele:

$$\begin{array}{rcrcll}
n=1 & : & \hat{y}_0 & = & y_0 & \\
n=2 & : & \hat{y}_0 & = & y_0+y_1 & \\
 & & \hat{y}_1 & = & y_0-y_1 & \\
n=4 & : & q & = & p^{-2\pi i/4}=-i & \\
 & & g_0 & = & y_0+y_2 & u_0=y_1+y_3 \\
 & & g_1 & = & y_0-y_2 & u_1=y_1-y_3 \\
 & & \hat{y}_0 & = & g_0+u_0 & \\
 & & \hat{y}_1 & = & g_1-iu_1 & \\
 & & \hat{y}_2 & = & g_0-u_0 & \\
 & & \hat{y}_3 & = & g_1+iu_1 &
\end{array}$$

Wir können die FFT auch als Faktorisierung der Matrix W_n für die Fourier-Transformation der Länge n ansehen, also

$$W_n = \left(e^{-2\pi i k\ell/n}\right)_{k,\ell=0,\ldots,n-1} \ .$$

Sei P_n die Permutationsmatrix

$$P_n = \begin{pmatrix} e_0 \\ e_2 \\ \vdots \\ e_{n-2} \\ e_1 \\ e_3 \\ \vdots \\ e_{n-1} \end{pmatrix}$$

mit dem Einheitsvektor e_j^* des C^n, in dem wir die Komponenten von 0 bis $n-1$ durchnummerieren. Sei $m = n/2$ und I_m die (m,m) Einheitsmatrix, D_m die Diagonalmatrix

$$D_m = \begin{pmatrix} 1 & & & \\ & q & & \\ & & \ddots & \\ & & & q^{m-1} \end{pmatrix} , \qquad q = e^{-2\pi i/n} .$$

Dann ist

$$W_n = \begin{pmatrix} I_m & I_m \\ I_m & -I_m \end{pmatrix} \begin{pmatrix} I_m & 0 \\ 0 & D_m \end{pmatrix} \begin{pmatrix} W_m & 0 \\ 0 & W_m \end{pmatrix} P_n .$$

Die FFT ergibt sich durch sukzessives Anwenden dieser Formel. Die Durchführung der FFT verlangt also nur drei Operationen:

1) Permutationen

2) Multiplikation mit q^ℓ (Twiddle factor)

3) Die Anwendung von $\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ (Butterfly)

5.5 Auswertung von Polynomen

Sei $p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + A_0$. Der Standard-Algorithmus zur Auswertung von p an der Stelle x ist

$$\begin{aligned} s_n &= a_n , \\ s_j &= x s_{j+1} + a_j \quad , \quad j = n-1, \ldots, 0 \\ p &= s_0 . \end{aligned}$$

Man kommt also mit n flops aus.

Ein entsprechender Algorithmus für trigonometrische Polynome ist der Algorithmus von Goertzel (1958). Er wertet die Ausdrücke

$$c = \sum_{k=0}^{n-1} y_k \cos k\varphi\ , \qquad s = \sum_{k=0}^{n-1} y_k \sin k\varphi$$

aus. Für $\varphi = 2\pi j/n$, $j = 0, \dots, n-1$, ist das ein Fall für FFT.

Satz 5.5.1 *Für* $\varphi \notin \pi\mathbb{Z}$, $n \in \mathbb{N}$ *sei*

$$\begin{aligned} u_j &= \frac{1}{\sin\varphi} \sum_{k=j}^{n-1} y_k \sin(k-j+1)\varphi \quad , \quad j = 1, \dots, n-1 \\ u_n &= u_{n+1} = 0\ . \end{aligned}$$

Dann gilt:

(a) $u_j = y_j + 2\cos\varphi\, u_{j+1} - u_{j+2} \quad , \quad j = n-1, n-2, \dots, 1$

(b) $\sum_{k=0}^{n-1} y_k \sin k\varphi = u_1 \sin\varphi$

(c) $\sum_{k=0}^{n-1} y_k \cos k\varphi = y_0 + \cos\varphi\, u_1 - u_2$

Beweis:

Zu **(a)**: Das Additionstheorem für den Sinus

$$\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$$

liefert uns die Formel

$$\sin(k-j\pm 1)\varphi = \sin(k-j)\varphi\cos\varphi \pm \cos(k-j)\varphi\sin\varphi\ .$$

Hiermit können wir (a) durch einfaches Nachrechnen zeigen:

$$\begin{aligned} & y_j + 2\cos\varphi\, u_{j+1} - u_{j+2} \\ &\quad = y_j + \frac{1}{\sin\varphi}\left\{ 2\cos\varphi \sum_{k=j+1}^{n-1} y_k \sin(k-j)\varphi - \sum_{k=j+2}^{n-1} y_k \sin(k-j-1)\varphi \right\} \end{aligned}$$

$$
\begin{aligned}
&= y_j + \frac{1}{\sin\varphi}\sum_{k=j+1}^{n-1} y_k \{2\cos\varphi\sin(k-j)\varphi - \sin(k-j-1)\varphi\} \\
&= y_j + \frac{1}{\sin\varphi}\sum_{k=j+1}^{n-1} y_k \{2\cos\varphi\sin(k-j)\varphi - (\sin(k-j)\varphi\cos\varphi - \cos(k-j)\varphi\sin\varphi)\} \\
&= y_j + \frac{1}{\sin\varphi}\sum_{k=j+1}^{n-1} y_k \{\cos\varphi\sin(k-j)\varphi + \sin\varphi\cos(k-j)\varphi\} \\
&= \frac{1}{\sin\varphi}\left\{ y_j \sin\varphi + \sum_{k=j+1}^{n-1} y_k \sin(k-j+1)\varphi \right\} \\
&= \frac{1}{\sin\varphi}\sum_{k=j}^{n-1} y_k \sin(k-j+1)\varphi \\
&= u_j
\end{aligned}
$$

(b) Dies folgt direkt aus der Definition von u_1.

(c) Es ist

$$
\begin{aligned}
y_0 + \cos\varphi u_1 - u_2 &= y_0 + \frac{\cos\varphi}{\sin\varphi}\sum_{k=1}^{n-1} y_k \sin k\varphi - \frac{1}{\sin\varphi}\sum_{k=2}^{n-1} y_k \sin(k-1)\varphi \\
&= y_0 + y_1\cos\varphi + \sum_{k=2}^{n-1} y_k \frac{\cos\varphi\sin k\varphi - \sin(k-1)\varphi}{\sin\varphi} \\
&= y_0 + y_1\cos\varphi + \sum_{k=2}^{n-1} y_k \cos k\varphi\ ,
\end{aligned}
$$

wobei wir zuletzt wieder das Additionstheorem benutzt haben.

□

Dies führt zu folgendem Programm:

goertzel (φ, y, n, s, c)
/⋆ Wertet für gegebene $y_0, \ldots, y_{n-1}$ die Zahlen c, s aus ⋆/
$\{u_n = u_{n+1} = 0;$
    $\gamma = \cos\varphi;$
    for $j = n-1,\ n-2, \ldots, 1$

$$
\begin{aligned}
&\quad u_j = y_j + 2\gamma u_{j+1} - u_{j+2};\\
&c = y_0 + u_1\gamma - u_2;\\
&s = u_1 \sin\varphi;
\end{aligned}
$$
}

Zahlenbeispiel: $n = 1001$, $\varphi = \frac{\pi}{2(n-1)}$, $y_j = \begin{cases} 1 & , \quad j = n-1 \\ 0 & , \quad \text{sonst} \,. \end{cases}$

Korrekt: $c = \cos\frac{\pi}{2} = 0$, $s = \sin\frac{\pi}{2} = 1$

Sun 4 mit float-Zahlen: $\tilde{c} = -0.012144$, $\tilde{s} = 0.992777$

Dies ist nicht befriedigend. Eine Erklärung findet sich im nächsten Paragraphen.

5.6 Differenzengleichungen

Eine lineare Differenzengleichung der Ordnung m mit konstanten Koeffizienten lautet

$$\sum_{\nu=0}^{m} u_{k+\nu}\alpha_\nu = f_k \ , \qquad k = 0, 1, \ldots \tag{6.1}$$

Wir setzen $\alpha_m \neq 0$ voraus.

Beispiele:

1) Die Formel von Goertzel (Satz 5.5.1), also

$$u_k = y_k + 2u_{k+1}\cos\xi - u_{k+2} \ .$$

2) Für die Tschebyscheff-Polynome $T_k(x)$ gilt

$$T_{k+1}(x) + T_{k-1}(x) = 2xT_k(x) \ , \quad T_0 = 1 \ , \quad T_1 = x \ .$$

3) Ebenfalls linear, aber nicht mit konstanten Koeffizienten ist die Differenzengleichung

$$J_{k+1}(x) + J_{k-1}(x) = \frac{2k}{x} J_k(x)$$

für die Bessel-Funktion $J_k(x)$ 1. Art der Ordnung k:

$$J_k(x) = \frac{1}{\pi}\int_0^\pi \cos(x\sin t - kt)dt$$

Die Differenzengleichung (6.1) heißt homogen, falls $f_k = 0$ ist für $k = 0, 1, \ldots,$ andernfalls inhomogen. Der homogenen Gleichung ordnen wir das charakteristische Polynom

$$p(\lambda) = \sum_{\nu=0}^{m} \alpha_\nu \lambda^\nu$$

zu.

Satz 5.6.1 *p habe die Nullstellen* $\lambda_1, \ldots, \lambda_r$ *mit den Vielfachheiten* $\sigma_1, \ldots, \sigma_r$, $\sigma_1 + \cdots + \sigma_r = m$. *Dann ist*

$$u_k = k^\sigma \lambda_\ell^k \ , \qquad 0 \le \sigma < \sigma_\ell \ , \quad \ell = 1, \ldots, r \tag{6.2}$$

Lösung der homogenen Gleichung (6.1), und jede Lösung der homogenen Gleichung (6.1) ist eine Linearkombination solcher Lösungen.

Beweis: Für $\sigma = 0$ ist

$$\begin{aligned} \sum_{\nu=0}^{m} u_{k+\nu}\alpha_\nu &= \sum_{\nu=0}^{m} \lambda_\ell^{k+\nu}\alpha_\nu = \lambda_\ell^k \sum_{\nu=0}^{n} \lambda_\ell^\nu \alpha_\nu \\ &= \lambda_\ell^k p(\lambda_\ell) = 0 \ . \end{aligned}$$

Ist λ_ℓ eine σ_ℓ-fache Wurzel von p, so ist für $0 \le \sigma < \sigma_\ell$

$$\left(\frac{d}{d\lambda}\right)^\sigma p(\lambda)\Big|_{\lambda=\lambda_\ell} = \sum_{\nu=0}^{m} \nu(\nu-1)\cdots(\nu-\sigma+1)\lambda_\ell^{\nu-\sigma}\alpha_\nu = 0 \ .$$

Es folgt

$$\sum_{\nu=0}^{m} \nu^\sigma \lambda_\ell^\nu \alpha_\nu = 0 \ , \quad \sigma = 0, \ldots, \sigma_\ell - 1$$

und damit

$$\sum_{\nu=0}^{m} u_{k+\nu}\alpha_\nu = \sum_{\nu=0}^{m} (k+\nu)^\sigma \lambda_\ell^{k+\nu}\alpha_\nu = 0 \ , \quad \sigma = 0, \ldots, \sigma_\ell - 1 \ .$$

Also sind (6.2) tatsächlich Lösungen der homogenen Gleichung (6.1).

Die Lösungen der homogenen Gleichung (6.1) bilden offenbar einen Vektorraum über $\mathbb{C}$. Jede Lösung u ist durch m Anfangswerte $u_0, \ldots, u_{m-1}$ eindeutig bestimmt. Es folgt, daß der Lösungsraum die Dimension m hat. Die m Lösungen (6.2) sind linear unabhängig und bilden daher eine Basis des Lösungsraumes.

□

Man nennt eine Differenzengleichung stabil, wenn es eine Konstante C gibt, so daß für jede Lösung der homogenen Gleichung

$$|u_k| \leq C \max_{0 \leq \ell < m} |u_\ell|$$

für alle $k \geq 0$ gilt. Sind dann für eine Lösung u_k der inhomogenen Gleichung $u_0, \ldots, u_{m-1}$ mit dem Fehler ε bekannt, so kann u_k für $k \geq m$ mit dem Fehler $\leq C\varepsilon$ berechnet worden.

Satz 5.6.2 *Die Differenzengleichung (6.1) ist genau dann stabil, wenn für* $\ell = 1, \ldots, r$ *gilt:*

$$|\lambda_\ell| < 1 \quad \textit{oder} \quad |\lambda_\ell| = 1 \quad \textit{und} \quad \sigma_\ell = 1 \; .$$

Beweis: Folgt sofort aus Satz 6.1.

□

Beispiele:

1) Bei der Formel von Goertzel ist $p(\lambda) = 1 - 2\lambda\cos\xi + \lambda^2$, also $\lambda_{1,2} = \cos\xi \pm i\sin\xi$. Die Rekursion ist stabil für $\xi \neq 0$, instabil für $\xi = 0$. Man hat Probleme für $\xi \sim 0$.

2) Es gibt Probleme bei $|x| \geq 1$.

3) Hier ist die Theorie nicht streng anwendbar, da die Differenzengleichung nicht konstante Koeffizienten hat. Aus dem Vergleich mit 2) vermutet man aber, daß man Probleme hat für $|k/x| \geq 1$.

Zahlenbeispiel: Wir wollen $J_6(1)$ berechnen, ausgehend von $J_0(1)$, $J_1(1)$:

$J_0(1)$	0.765198	exakt
$J_1(1)$	0.440051	exakt
$J_2(1)$	0.114904	1
$J_3(1)$	0.019565	2
$J_4(1)$	0.002486	9
$J_5(1)$	0.000323	73
$J_6(1)$	0.000744	723

Der Fehler in der letzten Spalte ist in Einheiten der letzten Stelle angegeben. Deutlich sieht man die Auswirkung der Instabilität.

5.7 Harmonische Analyse

Sei f 2π-periodisch. Solche Funktionen kann man in eine Fourier-Reihe entwickeln:

Satz 5.7.1 *Sei f stetig. Dann gilt*

$$
\begin{aligned}
f(x) &= \frac{a_0}{2} + \sum_{k=0}^{\infty} (a_k \cos kx + b_k \sin kx) , \\
a_k &= \frac{1}{\pi} \int_0^{2\pi} f(x) \cos kx dx \\
b_k &= \frac{1}{\pi} \int_0^{2\pi} f(x) \sin kx dx
\end{aligned}
$$

Beweis: Siehe etwa **Heuser** :*Lehrbuch der Analysis II.*

□

Bemerkungen:

1) Gilt sogar $f \in C^m$, so ist a_k, $b_k = \mathrm{O}(k^{-m})$.

2) Ist f stetig mit Ausnahme von x_0 und existieren dort die linken und rechten Grenzwerte $f(x_0 \pm 0)$, so konvergiert die Reihe in x_0 gegen $\frac{1}{2}\left(f(x_0 + 0) + f(x_0 - 0)\right)$.

3) Die Berechnung der a_k, b_k aus f heißt Fourier-Analyse, die Berechnung von f aus a_k, b_k heißt Fourier-Synthese. a_k, b_k heißen Fourier-Koeffizienten von f, die Reihe Fourier-Reihe von f.

Beispiel: $f(x) = \begin{cases} 1 & , \quad 0 \leq x < \pi \\ -1 & , \quad \pi \leq x < 2\pi \end{cases}$. Offenbar ist $a_k = 0$ für alle k und $b_k = 0$ für k gerade, $b_k = 4/\pi k$ für k ungerade. Also lautet die Fourier-Reihe

$$f(x) = \frac{4}{\pi}\left(\sin x + \frac{1}{3}\sin 3x + \frac{1}{5}\sin 5x + \cdots\right) .$$

Zur numerischen Fourier-Analyse nehmen wir an, daß f an den Stützstellen $x_j = 2\pi j/n$, $j = 0, \ldots, n-1$ gegeben ist. Dann hat man die Approximationen

$$a_k \sim a_{k,n} = \frac{2}{n}\sum_{j=0}^{n-1} f(x_j)\cos kx_j \ , \quad b_k \sim b_{k,n} = \frac{2}{n}\sum_{j=0}^{n-1} f(x_j)\sin kx_j \ .$$

Im Abschnitt über numerische Integration werden wir sehen, daß diese Näherungen (für 2π-periodisches f) kaum noch verbesserbar sind. Trotzdem verhalten sich die genäherten Fourier-Koeffizienten $a_{k,n}$, $b_{k,n}$ ganz anders als die exakten: $a_{k,n}$, $b_{k,n}$ haben in k die Periode n, während a_k, b_k für $k \to \infty$ gegen 0 streben. Man kann daher nur für $k \ll n$ erwarten, daß $a_{k,n}$, $b_{k,n}$ gute Näherungen für a_k, b_k darstellen.

Wie in §3 gehen wir zur komplexen Schreibweise und das Intervall $[-\pi, \pi]$ über und schreiben dann für Satz 1

$$\begin{aligned} f(x) &= \sum_{k=-\infty}^{+\infty} c_k \, e^{ikx} \ , \\ c_k &= \frac{1}{2\pi}\int_{-\pi}^{+\pi} f(x) \, e^{-ikx} dx \ . \end{aligned}$$

Die Approximation $c_{n,k}$ für c_k ist mit $x_j = \pi j/n, \ j = -n, \ldots, n-1$

$$c_{n,k} = \frac{1}{n}\sum_{j=-n}^{n-1} f_j \, e^{-ikx_j} = \frac{1}{n}\sum_{j=-n}^{n-1} f_j \, e^{-\pi ikj/n}$$

mit $f_j = f(x_j)$. Dies ist genau eine Fourier-Transformation der Länge $2n$. Die Fourier-Analyse kann also näherungsweise durch die schnelle Fourier-Transformation durchgeführt werden. Zur Fourier-Synthese verwenden wir aus den oben angedeuteten Gründen nur $c_{n,k}$ für $k = -n, \ldots, n-1$, d.h.

$$\begin{aligned} f(x_j) &\sim \sum_{k=-n}^{n-1} c_{n,k}\, e^{ikx_j} \\ &= \sum_{k=-n}^{n-1} c_{n,k}\, e^{\pi ikj/n} \ , \end{aligned}$$

weil $c_{n,k}$ in k die Periode n hat. Dies ist genau eine inverse diskrete Fourier-Transformation der Länge $2n$. Auch die Fourier-Synthese kann daher mit der schnellen Fourier-Transformation durchgeführt werden.

Um das asymptotische Verhalten der $c_{n,k}$ an das der c_k anzupassen, multipliziert man die $c_{n,k}$ noch mit "Abminderungsfaktoren" oder "Filterfaktoren" $d_{n,k}$ mit der Eigenschaft

$$\begin{aligned} d_{n,k} \sim 1 \quad &, \quad |k| \ll n \ , \\ d_{n,k} \sim 0 \quad &, \quad |k| \sim n \ . \end{aligned}$$

Eine häufige Wahl ist

$$d_{n,k} = \left(\frac{\sin k\pi/n}{k\pi/n} \right)^p$$

mit einem geeigneten p, etwa $p = 1, 2$ oder 3.

5.8 Splines

Ein Spline (spline (engl.) = Straklatte) ist ein dünner elastischer Stab, der zwischen Gewichte ("Knoten") gespannt Kurven darstellt. Sie wurden im Schiffsbau verwendet. Wir werden unten sehen, daß die Straklatte zwischen den Knoten durch Polynome beschrieben wird, die an den Knoten unter gewissen Differenzierbarkeitsbedingungen zusammengefügt werden. Unter Splines oder Splinefunktionen versteht man daher in der Numerik Funktionen, welche stückweise Polynome sind und die an den Nahtstellen gewissen Differenzierbarkeitsbedingungen genügen.

Definition 5.8.1 *Seien $x_0 < x_1 < \cdots < x_n$ reelle Zahlen. Eine Funktion s heißt Spline der Ordnung k zu $x_0, \ldots, x_n$, falls*

1) $s \in C^{k-2}(\mathbb{R}^1)$.

2) In jedem Intervall $[x_i, x_{i+1}]$ $i = -1, \ldots, n$ ist s ein Polynom vom Grade $\leq k-1$ $(x_{-1} = -\infty,\ x_{n+1} = +\infty)$.

(Für $k = 1$ ist 1) leer.)

Beispiel: $n = 0,\ x_0 = t$,

$$f(x) = (x-t)_+^{k-1} = \begin{cases} (x-t)^{k-1} & ,\quad x > t\ , \\ 0 & ,\quad \text{sonst}\ . \end{cases}$$

f ist ein Spline der Ordnung k zu x_0.

Mit Hilfe von f können wir bereits sehr nützliche Splines erzeugen. Sei $x_0 < x_1 < \cdots < x_n$. Wir setzen $f_i = f(x_i)$, $i = 0, \ldots, n$. Die f_i sind Funktionen von t, was wir aber in der Bezeichnung nicht zum Ausdruck bringen. Dann sind auch die dividierten Differenzen $[f_i, \ldots, f_{i+k}]$ Funktionen von t. Wir setzen

$$B_{i,k}(t) = (x_{i+k} - x_i)[f_i, \ldots, f_{i+k}]\ . \tag{8.1}$$

Beispiel: $k = 2$, $f(x) = (x-t)_+$. Es ist

$$\begin{aligned} B_{i,2}(t) &= (x_{i+2} - x_i)[f_i, f_{i+1}, f_{i+2}] \\ &= [f_{i+1}, f_{i+2}] - [f_i, f_{i+1}] \\ &= \frac{f_{i+1} - f_{i+2}}{x_{i+1} - x_{i+2}} - \frac{f_i - f_{i+1}}{x_i - x_{i+1}}\ . \end{aligned}$$

Wir betrachten 4 Fälle.

1) $t < x_i$. Dann ist $f_i = x_i - t$, $f_{i+1} = x_{i+1} - t$, $f_{i+2} = x_{i+2} - t$, und es wird $B_{i,2}(t) = 0$.

2) $x_i \leq t < x_{i+1}$. Jetzt ist $f_i = 0$, $f_{i+1} = x_{i+1} - t$, $f_{i+2} = x_{i+2} - t$, und es wird $B_{i,2}(t) = 1 - (x_{i+1} - t)/(x_{i+1} - x_i)$.

3) $x_{i+1} \leq t < x_{i+2}$. Jetzt ist $f_i = f_{i+1} = 0$, $f_{i+2} = x_{i+2} - t$, also $B_{i,2}(t) = (x_{i+2} - t)/(x_{i+2} - x_{i+1})$.

4) $x_{i+2} \leq t$. Es ist $f_i = f_{i+1} = f_{i+2} = 0$ und damit $B_{i,2}(t) = 0$.

Wir erhalten also die stückweise lineare stetige Funktion, die außerhalb $[x_i, x_{i+2}]$ verschwindet und die bei x_{i+1} den Wert 1 annimmt, also einen Spline der Ordnung 2.

Satz 5.8.1 *$B_{i,k}$ ist ein Spline der Ordnung k, der außerhalb $[x_i, x_{i+k}]$ verschwindet.*

Beweis: f_j ist für $t \neq x_j$ ein Polynom vom Grade $\leq k-1$ in t und für alle t $k-2$ mal stetig differenzierbar (für $k = 1$ ist die letzte Aussage leer). Als Linearkombination solcher Ausdrücke ist $[f_i, \ldots, f_{i+k}]$ in jedem Intervall $[x_j, x_{j+1}]$ $(j = i-1, \ldots, i+k)$ ein Polynom vom Grade $\leq k-1$ und $k-2$ mal stetig differenzierbar, also ein Spline der Ordnung k zu $x_i, \ldots, x_{i+k}$. Für $t < x_i$ ist $f_j = (x_j - t)^{k-1}$ ein Polynom vom Grade $k-1$ in x_j für $j = i, \ldots, i+k$. Die dividierte Differenz $[f_i, \ldots, f_{i+k}]$ ist daher 0. Für $t > x_{i+k}$ ist $f_j = 0$ für $j = i, \ldots, i+k$. In beiden Fällen ist $B_{i,k}(t) = 0$.

$\square$

Für das Weitere benötigen wir ein Hilfsmittel über dividierte Differenzen, das mit Splines eigentlich gar nichts zu tun hat.

Lemma 5.8.1 *(Leibniz'sche Formel)* *Sei $f_j = g_j h_j$, $j = i, \ldots, i+k$. Dann gilt*

$$[f_i, \ldots, f_{i+k}] = \sum_{r=i}^{i+k} [g_i, \ldots, g_r][h_r, \ldots, h_{i+k}] \, .$$

Beweis: Seien f, g, h die Interpolationspolynome vom Grade k zu den Stützstellen $x_i, \ldots, x_{i+k}$ und den Stützwerten f_j, g_j, h_j, $j = i, \ldots, i+k$. Das Newton'sche Interpolationspolynom für g, h lautet

$$\begin{aligned}
g(x) &= [g_i] + [g_i, g_{i+1}](x - x_i) + \cdots [g_i, \ldots, g_{i+k}](x - x_i) \cdots (x - x_{i+k-1}) \\
&= \sum_{r=i}^{i+k} [g_i, \ldots, g_r](x - x_i) \cdots (x - x_{r-1}) \, , \\
h(x) &= [h_{i+k}] + [h_{i+k}, h_{i+k-1}](x - x_{i+k}) + \cdots \\
&\qquad + [h_{i+k}, \ldots, h_i](x - x_{i+k}) \cdots (x - x_{i+1}) \\
&= \sum_{s=i}^{i+k} [h_s, \ldots, h_{i+k}](x - x_{i+k}) \cdots (x - x_{s+1}) \, .
\end{aligned}$$

Multiplikation ergibt

$$(gh)(x) = \sum_{r,s=i}^{i+k} [g_i, \ldots, g_r][h_s, \ldots, h_{i+k}](x-x_i) \cdots (x-x_{r-1})(x-x_{s+1}) \cdots (x-x_{i+k}) \, . \tag{8.2}$$

Wir teilen die Summe auf in die Summe für $r > s$ und den Rest. Die erste Summe verschwindet für $x = x_i, \ldots, x_{i+k}$. Die restliche Summe ist ein Polynom vom Grade $\leq k$, welches an den Stellen $x_i, \ldots, x_{i+k}$ die Werte $f_i, \ldots, f_{i+k}$ annimmt und daher mit f übereinstimmen muß.

Der Höchstkoeffizient des restlichen Polynoms ergibt sich als Summe über $r = s$, also

$$\sum_{s=i}^{i+k} [g_i, \ldots, g_s][h_s, \ldots, h_{i+k}] \, ,$$

und dies muß mit dem Höchstkoeffizienten des Newton'schen Interpolationspolynoms, also $[f_i, \ldots, f_{i+k}]$ übereinstimmen. Dies beweist die Leibniz'sche Formel.

□

Satz 5.8.2 *Für $k \geq 2$ und $i = 0, \ldots, n-k$ gilt*

$$B_{i,k}(x) = \frac{x - x_i}{x_{i+k-1} - x_i} B_{i,k-1}(x) + \frac{x_{i+k} - x}{x_{i+k} - x_{i+1}} B_{i+1,k-1}(x) \, .$$

Beweis: Nach Definition (5.8.1) ist

$$B_{i,k}(x) = (x_{i+k} - x_i)[f_i, \ldots, f_{i+k}] \ , \quad f_j = (x_j - x)_+^{k-1} \ .$$

Wir setzen $g_j = (x_j - x)_+^{k-2}$ und $h_j = x_j - x$. Dann ist $f_j = g_j h_j, \quad j = i, \ldots, i+k$ und damit nach Leibniz

$$B_{i,k}(x) = (x_{i+k} - x_i) \sum_{r=i}^{i+k} [g_i, \ldots, g_r][h_r, \ldots, h_{i+k}] \ .$$

Da h_j ein Polynom vom Grade 1 in x_j ist, verschwindet $[h_r, \ldots, h_{i+k}]$ für $r < i+k-1$, und wir erhalten

$$\begin{aligned} B_{i,k}(x) &= (x_{i+k} - x_i)\{[g_i, \ldots, g_{i+k}][h_{i+k}] + [g_i, \ldots, g_{i+k-1}][h_{i+k-1}, h_{i+k}]\} \\ &= (x_{i+k} - x_i)\{[g_i, \ldots, g_{i+k}](x_{i+k} - x) + [g_i, \ldots, g_{i+k-1}]\} \\ &= (x_{i+k} - x_i)\left\{\frac{[g_{i+1}, \ldots, g_{i+k}] - [g_i, \ldots, g_{i+k-1}]}{x_{i+k} - x_i}(x_{i+k} - x) + [g_i, \ldots, g_{i+k-1}]\right\} \\ &= [g_{i+1}, \ldots, g_{i+k}](x_{i+k} - x) + [g_i, \ldots, g_{i+k-1}](x - x_i) \\ &= \frac{B_{i+1,k-1}}{x_{i+k} - x_{i+1}}(x_{i+k} - x) + \frac{B_{i,k-1}(x)}{x_{i+k-1} - x_i}(x - x_i) \ . \end{aligned}$$

□

Bemerkungen:

1) Zusammen mit

$$B_{i,1}(x) = \begin{cases} 1 & , \quad x_i \le x < x_{i+1} \ , \\ 0 & , \quad \text{sonst} \end{cases}$$

ermöglicht Satz 5.8.2 die rekursive Berechnung der $B_{i,k}$.

2) Diese Rekursion ist numerisch stabil, da nur Linearkombinationen positiver Zahlen gebildet werden.

Satz 5.8.3 *Für $k \geq 3$ gilt*

$$B'_{i,k}(x) = (k-1)\left\{\frac{B_{i,k-1}(x)}{x_{i+k-1} - x_i} - \frac{B_{i+1,k-1}(x)}{x_{i+k} - x_{i+1}}\right\} .$$

Beweis: Nach Definition ist

$$B_{i,k}(x) = (x_{i+k} - x_i)[f_i, \ldots, f_{i+k}] , \quad f_j = (x_j - x)_+^{k-1} .$$

Differentiation nach x ergibt

$$B'_{i,k}(x) = -(k-1)(x_{i+k} - x_i)[g_i, \ldots, g_{i+k}] ,$$

$$g_j = (x_j - x)_+^{k-2} .$$

Nach der rekursiven Definition der dividierten Differenzen ist

$$\begin{aligned} B'_{i,k}(x) &= -(k-1)\{[g_{i+1}, \ldots, g_{i+k}] - [g_i, \ldots, g_{i+k-1}]\} \\ &= (k-1)\left\{\frac{B_{i,k-1}(x)}{x_{i+k-1} - x_i} - \frac{B_{i+1,k-1}(x)}{x_{i+k} - x_{i+1}}\right\} . \end{aligned}$$

□

5.9 Interpolation mit Splines

Sei $x_0 < x_1 < \cdots x_n$ und

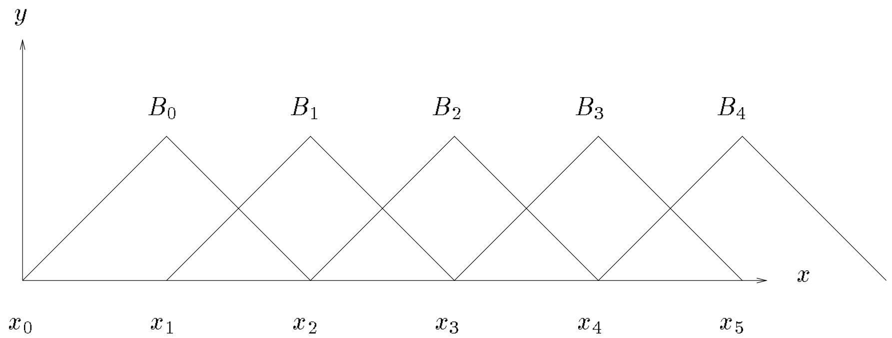
$$s(t) = \sum_{i=0}^{n-k} a_i B_{i,k}(t) \ .$$

Seien $n-k+1$ Stützstellen $t_0 < t_1 < \cdots < t_{n-k}$ und ebenso viele Stützwerte $y_0, \ldots, y_{n-k}$ gegeben. Wir wollen die a_i so bestimmen, daß $s(t_j) = y_j$, $j = 0, \ldots, n-k$ ist.

Beispiel: $k = 1$. Dann ist $s(t) = a_i$ für $x_i \leq t < x_{i+1}$. Das Interpolationsproblem ist genau dann eindeutig lösbar, wenn $x_i \leq t_i < x_{i+1}$, und zwar ist $a_i = y_i, \ \ i = 0, \ldots, n-k$.

Satz 5.9.1 *Das Interpolationsproblem ist eindeutig lösbar, wenn* $x_i < t_i < x_{i+k}, \ \ i = 0, \ldots, n-k$.

Beweis: Wir beginnen mit dem Fall $k = 2$ und schreiben B_i für $B_{i,k}$.



Wir haben zu zeigen, daß die $(n-1, n-1)$-Matrix

$$B = \begin{pmatrix} B_0(t_0) & B_1(t_0) & 0 & 0 \\ B_0(t_1) & B_1(t_1) & B_2(t_1) & 0 \\ 0 & B_1(t_2) & B_2(t_2) & B_3(t_2) \\ 0 & 0 & B_2(t_3) & B_3(t_3) \end{pmatrix} \qquad (n = 5)$$

invertierbar ist. Für $n = 2, 3$ ist dies elementar einzusehen. Den Fall $n > 3$ kann man auf $n = 2, 3$ zurückführen. Dazu zeigen wir, daß für $n > 3$ nicht alle Außerdiagonalelemente $\neq 0$ sein können. Ist nämlich etwa $B_0(t_1) \neq 0$, so ist $t_1 < x_2$ und damit $B_2(t_1) = 0$. Entsprechend argumentiert man in den anderen Fällen. Ist aber ein Außerdiagonalelement $= 0$, so zerfällt die Matrix in Teilmatrizen. Im Falle $B_2(t_1) = 0$ wären dies

$$\begin{pmatrix} B_0(t_0) & B_1(t_0) \\ B_0(t_1) & B_1(t_1) \end{pmatrix} \quad , \quad \begin{pmatrix} B_2(t_2) & B_3(t_2) \\ B_2(t_3) & B_3(t_3) \end{pmatrix} \; ,$$

im Falle $B_1(t_0) = 0$

$$B_1(t_0) \quad , \quad \begin{pmatrix} B_1(t_1) & B_2(t_1) & 0 \\ B_1(t_2) & B_2(t_2) & B_3(t_2) \\ 0 & B_2(t_3) & B_3(t_3) \end{pmatrix} .$$

Die letzte Matrix zerfällt wieder in eine $(1,1)$- und eine $(2,2)$-Matrix. Insgesamt genügt es also, die Invertierbarkeit der $(1,1)$- und der $(2,2)$-Matrizen nachzuweisen, also die Fälle $n = 2, 3$ zu betrachten. Damit ist der Satz für $k = 2$ bewiesen.

Sei nun die Behauptung richtig bis zur Ordnung $< k$ für ein $k \geq 2$. Der Fall $n = k$ ist trivial. Eine typische Stelle der Matrix für die Ordnung k sieht dann so aus:

$$\begin{array}{ccc} & B_j(t_{j-1}) & B_{j+1}(t_{j-1}) \\ B_{j-1}(t_j) & B_j(t_j) & B_{j+1}(t_j) \end{array}$$

Wäre hier $B_{j+1}(t_j) = 0$, d.h. $t_j \leq x_{j+1}$, so wären erst recht alle Elemente rechts und oberhalb dieses Elementes 0. Entsprechend würde aus $B_{j-1}(t_j) = 0$, d.h. $t_j \leq x_{j+k-1}$, folgen, daß alle Elemente links und unterhalb dieses Elementes 0 wären. In beiden Fällen zerfiele die Matrix in kleinere, und wir könnten Induktion nach n treiben. Wir können also annehmen, daß $x_{j+1} < t_j < x_{j+k-1}, \;\; j = 0, \ldots, n-k$.

Sei nun a ein Vektor mit den Komponenten $a_0, \ldots, a_{n-k}$ und $Ba = 0$. Der Spline

$$s = \sum_{j=0}^{n-k} a_j B_{j,k}$$

wäre eine C^{n-k}-Funktion mit den $n-k+3$ Nullstellen $t_0, \ldots, t_{n-k}, \;\; x_0, \; x_n$. Nach dem Satz von Rolle hätte s' $\; n-k+2$ Nullstellen

$$\tau_0 \in (x_0, t_0) \; , \quad \tau_{n-k+1} \in (t_{n-k}, x_n) \; ,$$
$$\tau_j \in (t_{j-1}, t_j) \; , \quad j = 1, \ldots, n-k \; .$$

Wegen $x_{j+1} < t_j < x_{j+k-1}$ muß dann

$$\tau_0 \in (x_0, x_{k-1}) \ , \quad \tau_{n-k+1} \in (x_{n-k+1}, x_n) \ ,$$
$$\tau_j \in (x_j, x_{j+k-1}) \ , \quad j = 1, \ldots, n-k$$

gelten, wie man sich an Hand einer Figur klarmacht.

Mit $\tau_0, \ldots, \tau_{n-k+1}$ haben wir $n-k+2$ Nullstellen von s' gefunden, die in den Trägern von $B_{0,k-1}, \ldots, B_{n-k+1,k-1}$ liegen. s' löst also das Interpolationsproblem für Splines der Ordnung $k-1$ mit den Interpolationsstellen $\tau_0, \ldots, \tau_{n-k+1}$ und den Werten 0. Nach Induktionsannahmen ist dieses Problem eindeutig lösbar. Eine Lösung ist offenbar $s' = 0$. Also ist $s' = 0$ und damit $s = 0$. Damit ist auch $a = 0$.

□

Empfohlenes Buch: J. Werner, Numerische Mathematik 1 + 2, Vieweg 1992, je DM 38,–

Weitere Literatur:

1 Neuere Lehrbücher

- **Deuflhard - Hohmann**: Numerische Mathematik. Eine algorithmisch orientierte Einführung. Walter de Gruyter 1991.
- **Hämmerlin - Hoffmann**: Numerische Mathematik. Springer 1989.
- **Stoer**: Numerische Mathematik I. Springer 1989.
- **Stoer - Bulirsch**: Numerische Mathematik II. Springer 1990.
- **Stummel - Hainer**: Praktische Mathematik. Teubner 1982.
- **Schwarz**: Numerische Mathematik. Teubner 1988.
- **Schaback - Werner, H.**: Numerische Mathematik. Springer 1992.

2 Ältere Lehrbücher

- **Acton**: Numerical Methods that Work. Harper 1970.
- **Björck - Dahlquist**: Numerische Methoden. Oldenbourg 1972.
- **Conte - de Boor**: Elementary Numerical Analysis. McGraw-Hill 1965, 1972.
- **Mennicken - Wagenführer**: Numerische Mathematik 1, 2, 3. Vieweg 1977.
- **Schmeisser - Schirmeier**: Praktische Mathematik. W. de Gruyter 1976.
- **Fröberg**: Introduction to Numerical Analysis. Addison-Wesley 1965.
- **Hamming**: Numerical Methods for Scientists and Engineers. McGraw-Hill, New York 1962.
- **Henrici**: Elements of Numerical Analysis. John Wiley & Sons, Inc., New York 1964.
- **Hildebrand**: Introduction to Numerical Analysis. McGraw-Hill, New York 1956.

- **Stiefel**: Einführung in die Numerische Mathematik. Teubner 1963.
- **Willers**: Methoden der praktischen Analysis. W. de Gruyter 1957.

3 Monographien

- **Beresin - Shidkow**: Numerische Methoden 1, 2. VEB, Deutscher Verlag der Wissenschaften, 1970.
- **Blum**: Numerical Analysis and Computation. Addison-Wesley & Sons, Inc., New York 1966.
- **Goldstine**: A History of Numerical Analysis. Springer 1977.
- **Hartree**: Numerical Analysis, 2d ed.. Oxford University Press, Fair Lawn, N.J. 1958.
- **Householder**: Principles of Numerical Analysis. McGraw-Hill, New York 1953.
- **Isaacson - Keller**: Analysis of Numerical Methods. John-Wiley & Sons, Inc., New York 1966.
- **Lanczos**: Applied Analysis. Prentice-Hall, Englewood Cliffs, N.J. 1956.
- **Marchuck**: Methods of Numerical Mathematics. Springer 1975.
- **Milne**: Numerical Calculus. Princeton University Press, Princeton, N.J. 1949.
- **Ralston**: A First Course in Numerical Analysis. McGraw-Hill 1965.
- **Young - Gregory**: A Survey of Numerical Mathematics. Addison-Wesley 1972.

4 Programmbibliotheken

- **Press-Flannery-Teukolsky-Vetterling**, Numerical Recipes in **C** (auch FORTRAN, PASCAL erhältlich), Cambridge University Press).
- **IMSL-Bibliothek** (International Mathematical & Statistical Libraries, Inc., Houston).
- **NAG-Bibliothek** (The Numerical Analysis Group Ltd., Oxford).