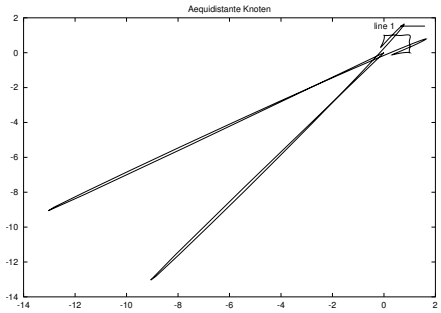
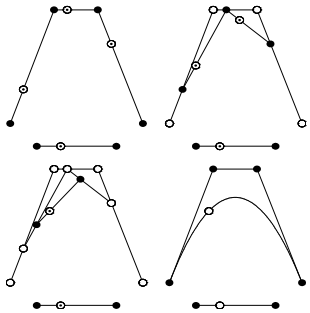


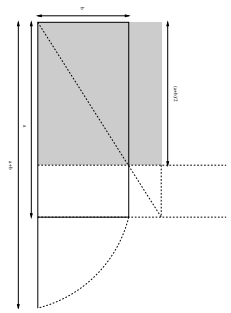


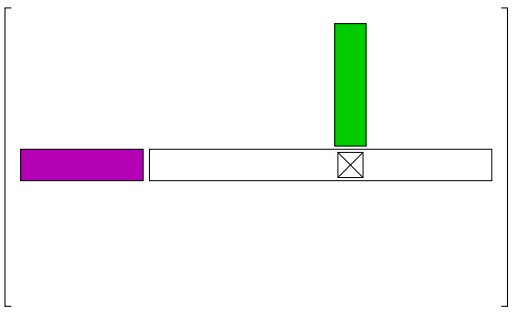
Numerische Mathematik I

Vorlesung, zuerst gehalten im Wintersemester 1999/2000

Tomas Sauer

Version 3.1
Letzte Änderung: 14.4.2011





Statt einer Leerseite . . . 0

Es gibt sogenannte Mathematiker, die sich gerne ebenso für Gesandte der Weisheit gehalten wissen wissen möchten als manche Theologen für Gesandte Gottes und ebenso das Volk mit algebraischem Geschwätz, das sie Mathematik nennen, als jene mit einem Kauderwelsch hintergehen, dem sie den Namen biblisch beilegen.
[. . .]
Dieses ist so gewiß, als $(a - x) \cdot (a + x) = a^2 - x^2$ ist.

Georg Christoph Lichtenberg

0 Inhaltsverzeichnis

To solve a well–conditioned problem with an ill–conditioned method is a numerical crime.

W. Gautschi

1 Was will Numerische Mathematik?

Viele Probleme aus der "realen" Welt lassen sich mathematisch formulieren und dann entweder von Hand oder unter Zuhilfenahme eines Computers lösen. Dabei stellt man im allgemeinen die folgenden Forderungen an ein Lösungsverfahren:

Interpretierbarkeit: Ein potentieller Anwender will die Lösung interpretieren können. So ist beispielsweise π oder `RootOf( x^13 - 12 x^7 + 9 x^3 - 1 )` in vielen Fällen nicht ausreichend.

Effizienz: Die Lösung soll innerhalb einer bestimmten problemabhängigen Zeitspanne (Echtzeit?) geliefert werden.

Genauigkeit: Die berechnete Lösung soll innerhalb einer bestimmten Toleranzschranke um die gewünschte Lösung liegen.

Um die ersten beiden Forderungen, also Interpretierbarkeit und Verfügbarkeit gewährleisten zu können, muß man sich meistens mit einer *Näherung* an die exakte Lösung zufriedengeben. Numerische Mathematik beschäftigt sich mit

- der Konstruktion von Verfahren zum Auffinden von "Lösungen".
- der Analyse dieser Verfahren bezüglich *Effizienz* und *Störungsanfälligkeit*.

Oftmals gibt es natürlich mehrere Lösungsverfahren für ein Problem, die zudem für einen Problemtyp sehr gut, für einen anderen sehr schlecht sein können. Natürlich gibt es auch Verfahren, die immer schlecht sind.

Außerdem sind viele "Verfahren", die man so in "reinen" Mathematikvorlesungen kennenlernt, in der Praxis mit einiger Vorsicht zu genießen. Wir werden uns mal ein paar Beispiele ansehen.

1.1 Berechnung der Ableitung

Die Ableitung einer Funktion $f \in C^1(\mathbb{R})$ ist bekanntlich definiert als

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}, \qquad x \in \mathbb{R}.$$

```
%%  Diff1.m (Numerik I)
%%  ---------------------------------------------------
%%  Berechnung der Ableitung
%%  Eingabe:
%%    fn   Name der Funktion
%%    x    Stelle
%%    N    Schrittweite 2^(-N)
%%  Nur fuer Octave!

function res = Diff1( fn,x,N )
  h = 1;

  for i = 0:N
    res = ( feval( fn,x+h ) - feval( fn,x ) ) / h;
    h = h/2;
    printf( "2^{-%d} : %11e\n", i, res );
  end
%endfunction
```

Programm 1.1 `Diff1.m`: Differentiation

Um diesen Grenzwert auszurechnen, verwenden wir `Diff1.m`. Tatsächlich sieht man aber beim Aufruf

```
Diff1( "cos", 1, 55 );
```

daß sich die Werte zuerst dem korrekten Wert `-.841471` annähern, dann aber wieder davon entfernen und schließlich `.00000` werden und auch bleiben. Es ist also von entscheidender Bedeutung, mit dem Limes früh genug aufzuhören.

1.2 Lösung eines linearen Gleichungssystems

Wir betrachten eine Matrix $A \in \mathbb{R}^{n\times n}$ sowie $b \in \mathbb{R}^n$ und suchen nach $x \in \mathbb{R}^n$, so daß $Ax = b$. In der Linearen Algebra lernt man ein "Verfahren", um x zu berechnen, nämlich die *Cramersche Regel*. Dazu definieren wir die Matrizen

$$A_j(b) = \begin{bmatrix} a_{11} & \dots & a_{1,j-1} & \boxed{b_1} & a_{1,j+1} & \dots & a_{1n} \\ \vdots & \ddots & \vdots & \boxed{\vdots} & \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{n,j-1} & \boxed{b_n} & a_{n,j+1} & \dots & a_{nn} \end{bmatrix}.$$

Satz 1.1 *Die Lösung x von $Ax = b$ ergibt sich als*

$$x_j = \frac{\det A_j(b)}{\det A}, \qquad j = 1, \dots, n. \tag{1.1}$$

Die Determinanten können wir nun nach der Leibniz–Regel berechnen:

$$\det A = (-1)^j \sum_{k=1}^{n} (-1)^k \, a_{jk} \det A_{jk}, \qquad j = 1, \ldots, n,$$

wobei

$$A_{jk} = \begin{bmatrix} a_{11} & \ldots & a_{1,k-1} & a_{1,k+1} & \ldots & a_{1n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{j-1,1} & \ldots & a_{j-1,k-1} & a_{j-1,k+1} & \ldots & a_{j-1,n} \\ a_{j+1,1} & \ldots & a_{j+1,k-1} & a_{j+1,k+1} & \ldots & a_{j+1,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \ldots & a_{n,k-1} & a_{n,k+1} & \ldots & a_{nn} \end{bmatrix}.$$

diejenige Matrix ist, die man aus A durch Streichung der j–ten Zeile und der k–ten Spalte erhält. Das führt zu folgendem "Algorithmus":

1. Berechne $a = \det A$.

2. Ist $a = 0$, dann ist das Problem unlösbar, ist $a \neq 0$ berechne die Werte x_j, $j = 1, \ldots, n$, nach (1.1).

Zu dieser "Lösungsmethode" kann man einiges sagen:

- Der Test $a = 0$ setzt voraus, daß $\det A$ in einer exakten Arithmetik berechnet wird, denn ansonsten kann man sich fast sicher sein, daß durch Rundungsfehler der Wert leicht verändert und somit $\neq 0$ wird.

- Die Determinantenberechnung ist sehr aufwendig: sei $f(n)$ die Anzahl der Rechenoperationen, die man für die Berechnung einer $n \times n$ Determinante benötigt, dann sagt uns die Leibniz–Regel, daß $f(2) = 3$ und $f(n) = 2nf(n-1)$, also

  $$f(n) = \frac{3}{8} 2^n n!$$

  Wir werden später sehen, daß "vernünftige" Vefahren zur Berechnung der Lösung Cn^3 Rechenoperationen benötigen.

  Dies wird besonders heftig, wenn man die Determinante in exakter Arithmetik (z.B. in MAPLE) berechnet.

- Die Berechnung von Determinanten ist *extrem* anfällig gegen kleine Störungen der Elemente, also instabil

Wenn man also ein Problem praktisch lösen will, ist es, im Gegensatz zur "theoretischen" Mathematik wirklich wichtig, *wie* man es macht.

1.3 Berechnung der Varianz

Die Berechnung von Mittelwerten und Varianzen ist eine häufig auftretende "Arbeit" in der Statistik, die so monoton und stupide ist, daß man sie besser dem Computer überläßt. Dabei kann man aber durchaus einiges an Fehlern einbauen, wenn man sich nicht auch um die Numerik kümmert.

Definition 1.2 *Es sei* $X = (x_1, \ldots, x_N) \in \mathbb{R}^N$ *ein Vektor von Zahlen (Meßwerten ...). Der* Erwartungswert $E(X)$ *ist definiert als*

$$E(X) = \frac{1}{N} \sum_{j=1}^{N} x_j, \tag{1.2}$$

und die (empirische) Standardabweichung *als*

$$S(X) = \sqrt{\frac{1}{N-1} \sum_{j=1}^{N} (x_j - E(X))^2}. \tag{1.3}$$

Die Größe $V(X) = S^2(X)$ *bezeichnet man als* Varianz.

Das Problem bei der naiven Berechnung der Varianz besteht nun darin, daß man die Meßwerte *zweimal* durchlaufen müsste, zuerst bei der Berechnung des Erwartungswerts und dann bei der Berechnung der Abweichung vom Erwartungswert. Dies würde es nötig machen, die Meßwerte zwischenzuspeichern, was bei vielen, vielen Meßwerten ein zusätzlicher Programm- und Speicheraufwand wäre. Daher sucht man nach Methoden, die mit nur *einem* Durchlauf durch die Daten auskommen.

In vielen Statistikbüchern findet man die folgende, sehr ansprechende Formel:

$$V(X) = \frac{1}{N-1} \left(\sum_{j=1}^{N} x_j^2 - \frac{1}{N} \left(\sum_{j=1}^{N} x_j \right)^2 \right). \tag{1.4}$$

Damit ergibt sich die Berechnungsvorschrift

$$S_0 = Q_0 = 0 \tag{1.5}$$

$$S_j = S_{j-1} + x_j,\ Q_j = Q_{j-1} + x_j^2, \qquad j = 1, \ldots, n \tag{1.6}$$

und

$$S(X) = \sqrt{\frac{1}{N-1} \left(Q_N - \frac{S_N^2}{N} \right)}, \tag{1.7}$$

Übung 1.1 Beweisen Sie die Formel (1.4).

So schön diese Formel ist, so schön kann man damit auf die Nase fallen.

```
%%  Varianz1.m (Numerik 1)
%%  ------------------------------------------------------
%%  One-pass Varianzberechnung auf die 'dumme' Art
%%  Eingabe:
%%    x    Vektor

function V = Varianz1( x )
  N = length( x );
  S = 0;
  Q = 0;

  for i = 1:N
    S = S + x(i);
    Q = Q + x(i)^2;
  end

  V = sqrt( ( Q - S^2 / N ) / (N-1) );
%endfunction
```

Programm 1.2 `Varianz1.m`: Varianzberechnung nach (1.5), (1.6) und (1.7).

Beispiel 1.3 *(C–Programm* `Var.c`*) Um die Effekte richtig sichtbar zu machen verwenden wir ein C–Programm, bei dem in* einfacher Genauigkeit *(*`float`*) gerechnet wird.*

1. *Wir betrachten* $X = (5000, 5001, 5002)$*. Man sieht leicht, daß* $E(X) = 5001$ *und* $S(X) = 1$*. Das Programm liefert aber*[1] *den ziemlich falschen Wert* $S(X) = 0.707107$*.*
2. *Noch besser wird es für* $X = (10000, 10001, 10002)$*: Das Ergebnis wird dann* `NaN` *(not a number), d.h., es gibt keine Standardabweichung mehr oder diese ist eine komplexe Zahl*[2] ...

Für ein stabileres Verfahren setzen wir $X_k = (x_1, \ldots, x_k)$ und bestimmen die Werte

$$M_k = E(X_k) \qquad \text{und} \qquad Q_k = (k-1)V(X_k)^2 \tag{1.8}$$

mittels der Berechnungsregeln $M_0 = Q_0 = 0$ und

$$Q_j = Q_{j-1} + \frac{(j-1)(x_j - M_{j-1})^2}{j}, \qquad j = 1, \ldots, n, \tag{1.9}$$

$$M_j = M_{j-1} + \frac{x_j - M_{j-1}}{j}, \qquad j = 1, \ldots, n. \tag{1.10}$$

[1] Auf (m)einem Pentium II unter Linux, SuSE Version 6.0 – also schon ein klein wenig veraltet in Zwischenzeit.
[2] Was ein klein wenig im Gegensatz zu (1.3) steht, wo lediglich nichtnegative Zahlen aufsummiert werden.

```
/*
  Var.c
  Berechnung der Varianz einer Folge

  Compile:      gcc -o Var Var.c -lm
*/

#include <stdio.h>
#include <math.h>

float sqr( float x ) { return x*x; }

float Var( unsigned N, float *data )
{
  float s1 = 0, s2 = 0;
  unsigned n;

  for ( n = 0; n < N; n++ )
  { s1 += sqr( data[n] ); s2 += data[n]; }

  s1 -= sqr( s2 ) / N; s1 /= (float)( N - 1);

  return (float)sqrt( s1 );
}

main()
{
  static float data[1111];
  unsigned N,n;

  /* Einlesen */

  scanf( "%d", &N );
  for ( n = 0; n < N; n++ ) scanf( "%f", data + n );

  /* Ausgeben */

  printf( "Var = %f / %f\n", Var( N, data ) );
}
```

Programm 1.3 `Var.c`: Das erste, letzte und einzige C–Programm. Es berechnet die Varianz nach (1.5), (1.6) und (1.7) in *einfacher* Genauigkeit.

```
%%  Varianz2.m (Numerik 1)
%%  ------------------------------------------------------
%%  One-pass Varianzberechnung auf die 'clevere' Art
%%  Eingabe:
%%    x    Datenvektor

function V = Varianz2( x )
  N = length( x );
  M = Q = 0;

  for i = 1:N
    Q = Q + ( i - 1 ) * ( x(i) - M )^2 / i;
    M = M + ( x(i) - M ) / i;
  end

  V = sqrt( Q / (N-1) );
endfunction
```

Programm 1.4 `Varianz2.m`: Varianzberechnung nach (1.9), (1.10) und (1.11).

Das Ergebnis ist dann

$$V(X) = \sqrt{\frac{Q_N}{N-1}}. \tag{1.11}$$

Dieses Schema berechnet die Variation aus Beispiel 1.3 korrekt zu $V(X) = 1.00000$.

Übung 1.2 Beweisen Sie, das M_k und Q_k aus (1.8) tatsächlich den Rekursionsformeln aus (1.9) und (1.10) genügen.

Satz 1.4 *Numerik ist zu wichtig, als daß man sie den Statistikern überlassen sollte.*

1.4 Aber sowas passiert doch nicht wirklich . . .

Beispiel 1.5 *(Börse von Vancouver)*
Der Aktienindex der Börse von Vancouver wurde in den frühen 80ern mit Computern berechnet. Im Januar 1982 wurde der Aktienindex mit dem Wert 1000 *gestartet und landete im November 1983 bei einem Stand von* 520*, obwohl allem Anschein die individuellen Aktien eigentlich ganz gut standen. Grund: Der Index wurde auf drei Dezimalstellen berechnet und dann wurde* abgeschnitten *anstatt zu runden. Dieser Index wurde (wohl aus dem "alten" Index und den individuellen Differenzen der Aktien) mehrere tausend Mal pro Tag upgedated und jedes Mal wurde abgeschnitten anstatt zu runden. Nach einer Neuberechnung mit Rundung* verdoppelte *sich der Aktienindex beinahe.*

Unwissenheit ist der schlimmste Fehler.

Persisches Sprichwort

2 Fehleranalyse

Eine besonders schöne, elementare und lesenswerte Einführung in das Wesen und Unwesen der Rundungsfehler und deren Konsequenzen ist die Arbeit von Goldberg [17] – aus ihr stammt auch der Beweis der Kahan–Summation (Satz 2.25).

Zuerst wollen wir uns aber mit den grundlegenden Problemen beim Rechnen mit *endlicher* Genauigkeit auf heutigen (digitalen) Computern befassen. Dabei werden zwei Typen von Fehlern untersucht.

Definition 2.1 *Es sei c eine Zahl und $\hat{c}$ die berechnete Größe.*

1. *Der* absolute Fehler *ist der Wert*

$$e_{\text{abs}} = e_a := |\hat{c} - c| \,,$$

2. *der* relative Fehler *(für $c \neq 0$) ist*

$$e_{\text{rel}} = e_r := \frac{|\hat{c} - c|}{|c|}.$$

Wie der Name schon sagt, gibt uns der absolute Fehler die *absolute* Genauigkeit des Ergebnisses an, während uns der relative die Anzahl der korrekten Ziffern im Ergebnis angibt.

Übung 2.1 Zeigen Sie: wird die B–adische Zahl $x = .x_1x_2\cdots x_\infty \times B^e$, $x_j \in \{0,\ldots,B-1\}$, $e \in \mathbb{Z}$, durch $\hat{x} = .\hat{x}_1\hat{x}_2\cdots\hat{x}_\infty \times B^e$, $\hat{x}_j \in \{0,\ldots,B-1\}$, mit einem relativen Fehler von ε angenähert, so stimmen praktisch[3] immer die ersten $-\log_B \varepsilon$ Stellen von x und $\hat{x}$ überein.

2.1 Gleitkommazahlen und -operationen

Alle heutigen Rechner verwenden eine sogenannte *Gleitkomma-, Fließpunkt-, Gleitpunkt-* oder *Fließkommaarithmetik* (engl. “floating point”).

[3] Es gibt ein paar mehr oder weniger pathologische Ausnahmen – diese kann man sich im Rahmen eines Lösungsversuchs ja herleiten!

Definition 2.2 *Die Menge* $\mathbb{F} = \mathbb{F}(B, m, E)$ *von* Gleitkommazahlen *zur* Basis $B \in \mathbb{N}$*, mit* Mantissenlänge $m \in \mathbb{N}$ *und* Exponentenbereich $E \subset \mathbb{Z}$ *besteht aus allen Zahlen der Form*

$$\pm \left(\sum_{j=1}^{m} d_j B^{-j} \right) B^e, \qquad d_j \in \{0, \ldots, B-1\}, \, e \in E.$$

Wir schreiben diese Zahlen auch als

$$\pm . \, d_1 \cdots d_m \;\times\; B^e.$$

Die Ziffern $d_1 \cdots d_m$ *bezeichnet man auch als* Mantisse.

Normalerweise sind Gleitkommazahlen *binär*, das heißt, $B = 2$. Schematisch sehen dann diese Zahlen so aus:

$\pm$	$d_1 \cdots d_m$	$e_1 \cdots e_k$
↑	↑	↑
Vz	Mantisse	Exponent

Der Exponent wird in einem "ganz normalen" vorzeichenbehafteten Ganzzahlformat (Einer- oder Zweierkomplement?) dargestellt. Achtung: damit wird der Exponentenbereich normalerweise nicht symmetrisch sein.

Bemerkung 2.3

1. *In obiger Darstellung kann es die Zahlen* $+0$ *und* -0 *geben und diese können sehr wohl* unterschiedlich *sein.*

2. *Für die* Größe *des Zahlenbereiches ist die Länge des Exponenten verantwortlich, für die* Genauigkeit *der Rechenoperationen hingegen die Mantissenlänge.*

3. *Der darstellbare*[4] *Bereich von* $\mathbb{F}$ *ist das Intervall*

$$D(\mathbb{F}) := \left[-B^{\max E}, -B^{\min E - 1}\right] \cup \left[B^{\min E - 1}, B^{\max E}\right] \cup \{0\}.$$

   Entsteht beispielsweise durch Rechnung eine Zahl x *mit* $|x| > B^{\max E}$*, dann spricht man von* Überlauf *(diese Zahlen haben den Wert* ∞*), bei* $|x| < B^{\min E - 1}$ *spricht man hingegen von* Unterlauf.

Definition 2.4 *Eine Gleitkommazahl* $x = .d_1 \cdots .d_m \times B^e \in \mathbb{F}$ *heißt* normalisiert*, wenn* $d_1 \neq 0$.

Eigentlich sind normalisierte Gleitkommazahlen auch die "vernünftigen" Gleitkommazahlen: Wenn man schon nur endliche Genauigkeit zur Verfügung hat, macht es ja nur wenig Sinn, Ziffern damit zu vergeuden, daß man führende Nullen mitschleppt.

[4] So ist die Aussage nicht ganz richtig: es ist möglich betragsmäßig *kleinere* Zahlen darzustellen, nämlich bis hin zu $\pm B^{\min E - m}$. Solche Zahlen, deren erste Ziffer 0 ist und bei denen der Exponent gleichzeitig minimal ist, bezeichnet man als *subnormal*. Allerdings würde eine detailliertere Betrachtung unter Berücksichtigung subnormaler Zahlen die Darstellung verkomplizieren, ohne viel zu bewirken.

Definition 2.5 *Die Zahl* $u = \frac{1}{2}B^{1-m}$ *heißt* Rundungsfehlereinheit *(unit roundoff).*

Der Name *Rundungsfehlereinheit* kommt daher, daß dies der relative Fehler ist, der auftritt, wenn man von einer darstellbaren rellen Zahl x zur nächstgelegenen Fließkommazahl $\hat{x} \in \mathbb{F}$ übergeht.

Proposition 2.6 *Für jede Zahl* $x \in D(\mathbb{F})$ *gibt es eine normalisierte Gleitkommazahl* $\hat{x} \in \mathbb{F}$ *und ein* $\delta \in [-u, u]$*, so daß*

$$\hat{x} = x(1+\delta).$$

Beweis: Wir schreiben x in seiner normalisierten B–adischen Entwicklung als

$$x = \pm B^e \left(\sum_{j=1}^{\infty} x_j B^{-j} \right) = \pm .x_1 x_2 \cdots \times B^e,$$

wobei $x_1 \neq 0$, also $B^{e-1} \leq |x| \leq B^e$. Ohne Einschränkung können wir außerdem annehmen, daß $x > 0$ ist.

Es seien

$$x_\downarrow := .x_1 \cdots x_m \times B^e \in \mathbb{F} \qquad \text{und} \qquad x_\uparrow := \left(.x_1 \cdots x_m + B^{-m}\right) \times B^e \in \mathbb{F},$$

dann ist $x_\downarrow \leq x \leq x_\uparrow$ und einer der beiden Werte $|x - x_\downarrow|$ und $|x - x_\uparrow|$ ist $\leq \frac{1}{2}B^{e-m}$, denn sonst wäre ja

$$B^{e-m} = x_\uparrow - x_\downarrow = (x_\uparrow - x) + (x - x_\downarrow) > B^{e-m}.$$

Wählt man $\hat{x} \in \{x_\downarrow, x_\uparrow\}$ passend, dann ist also

$$\frac{|x - \hat{x}|}{|x|} \leq \frac{1}{2}\frac{B^{e-m}}{B^{e-1}} = \frac{1}{2}B^{1-m}.$$

Für einen alternativen Beweis für *gerade*[5] Werte von B setzen wir

$$y := \left\{ \begin{array}{ll} x, & x_{m+1} < \frac{B}{2}, \\ x + \frac{B^{e-m-1}}{2}, & x_{m+1} \geq \frac{B}{2}. \end{array} \right\} = y_0.y_1 \cdots y_m \cdots y_\infty \times B^e.$$

Dabei kann es zwar passieren, daß $y_0 = 1$ ist (z.B. beim Aufrunden von $.995$ auf zwei Stellen), aber dann muß $y_1 = \cdots = y_m = 0$ sein. Nun setzen wir

$$\hat{x} := \left\{ \begin{array}{ll} .y_1 \cdots y_m \times B^e & y_0 = 0, \\ .y_0 \cdots y_{m-1} \times B^{e+1} & y_0 = 1, \end{array} \right.$$

[5] Man kann sich relativ leicht klarmachen, warum die Basis eine gerade Zahl sein muß: Nachdem man ja immer die B Ziffern $0, \ldots, B-1$ zur Verfügung hat, rundet man im Falle einer geraden Basis gerade die Hälfte der Ziffern auf und die Hälfte der Ziffern ab, ist hingegen B ungerade, so muß man bei Auftreten der Ziffer $\frac{B-1}{2}$ die Entscheidung, ob auf- oder abgerundet werden soll, auf die nächste Ziffer "vertagen". Und wenn die wieder $\frac{B-1}{2}$ ist ...

(Achtung: $\hat{x} = .y_0 \cdots y_m \times B^{e+1}$ da $y_m = 0$!). Dann ist

$$\hat{x} = x + \left\{ \begin{array}{ll} -x_{m+1}B^{e-m-1}, & x_{m+1} < \frac{B}{2} \\ (B/2 - x_{m+1})\, B^{e-m-1}, & x_{m+1} \geq \frac{B}{2} \end{array} \right\} - \sum_{j=m+2}^{\infty} x_j\, B^{e-j}.$$

Also definieren wir

$$\varepsilon_{m+1} := \left\{ \begin{array}{ll} x_{m+1} & x_{m+1} < \frac{B}{2}, \\ x_{m+1} - \frac{B}{2} & x_{m+1} \geq \frac{B}{2}, \end{array} \right.$$

und da B gerade ist, ergibt sich

$$\varepsilon_{m+1} < \frac{B}{2} \qquad \Longrightarrow \qquad \varepsilon_{m+1} \leq \frac{B}{2} - 1$$

und somit

$$\begin{aligned} |x - \hat{x}| &= B^e \left(B^{-m-1}\varepsilon_{m+1} + \sum_{j=m+2}^{\infty} x_j B^{-j} \right) \\ &\leq B^e B^{-m-1} \left(\left(\frac{B}{2} - 1 \right) + (B-1)\frac{B^{-1}}{1 - B^{-1}} \right) \\ &= \frac{1}{2} B^{e-1} B^{1-m} \leq \frac{1}{2}|x| B^{1-m} = u|x|. \end{aligned}$$

Das Interessante am zweiten Beweis ist die Tatsache, daß wir zur Bestimmung von $\hat{x}$ nur die ersten $m+1$ Ziffern von x kennen müssen[6]. □

Übung 2.2 Zeigen Sie: Bei Verwendung einer *ungeraden* Basis B kann man unter Verwendung von $m+1$ Ziffern *nicht* mit der gewünschten Genauigkeit u auf m Stellen runden. Funktioniert es mit $m+k$ Stellen für irgendein $k > 1$?

Definition 2.7 *Die Gleitkommaoperationen*

$$\{\oplus, \ominus, \otimes, \oslash\} : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$$

werden wie folgt ausgeführt:

1. *Addition ($\oplus, \ominus$): Schiebe die Zahl mit kleinerem Exponenten so lange nach rechts[7], bis die beiden Exponenten gleich groß sind, führe dann die Operation aus, normalisiere das Ergebnis und runde auf die nächste Zahl aus $\mathbb{F}$.*

2. *Multiplikation ($\otimes, \oslash$): Addiere/subtrahiere die Exponenten, führe die Operation hinreichend exakt aus und normalisiere.*

Übung 2.3 Zeigen Sie, daß die Gleitkommaoperationen nicht mehr den Körperaxiomen genügen.

[6] Und das funktioniert eben nur, wenn die Basis B eine gerade Zahl ist

[7] Diese Operation erhöht den Exponenten.

Bemerkung 2.8

1. *Die obige Definition legt* nicht *fest, in welchem Format die Operationen exakt gerechnet werden (Akkumulator).*

2. *Die naive Anschauung "Führe die arithmetische Operation exakt durch und runde dann" ist natürlich völlig unrealistisch. Trotzdem werden wir sehen, daß sie unter gewissen (einfachen) Annahmen an die Arithmetik gemacht werden kann.*

3. *Die kritischste Operation ist die Subtraktion, die zeitaufwendigste die Division (bis zu Faktor 20 langsamer).*

4. *Ist* $x = .d_1 \cdots d_m B^e$ *und* $y = .\hat{d}_1 \cdots \hat{d}_m B^{\hat{e}}$ *dann fällt bei der Addition der betragsmäßig kleinere Operand unter den Tisch falls* $|e - \hat{e}| > m$.

Beispiel 2.9 *Sei* $B = 10$ *und* $m = 3$.

1. $.123 \times 10^0 \oplus .306 \times 10^{-1}$, *mindestens vierstelliger Akkumulator:*

$$\begin{array}{rr} & .1230 \\ + & .0306 \\ \hline & .1536 \end{array} \quad \rightarrow \quad .154$$

2. $.123 \times 10^0 \oplus .306 \times 10^{-1}$, *dreistelliger Akkumulator:*

$$\begin{array}{rr} & .123 \\ + & .030 \\ \hline & .153 \end{array} \quad \rightarrow \quad .153$$

3. $.123 \times 10^0 \ominus .122 \times 10^0$ *ergibt*

$$\begin{array}{rr} & .123 \\ - & .122 \\ \hline & .001 \end{array} \quad \rightarrow \quad .100 \times 10^{-2}$$

   Die letzten beiden Ziffern sind Phantasieziffern*! Diesen Vorgang bei der Subtraktion bezeichnet man als* Auslöschung.

Übung 2.4 Zeigen Sie: die Multiplikation zweier normierter Gleitkommazahlen liefert wieder eine normierte Gleitkommazahl.

Beispiel 2.10 *(IEEE 754) Die in heutigen PC–Prozessoren verwendete Gleitkommaarithmetik entspricht dem* IEEE[8] 754*–Standard. Die dabei verwendete Basis ist* 2 *und die arithmetischen Datentypen sind als*

[8] IEEE = Institute of Electrical and Electronical Engineers

Typ	*Mantisse*	*Exponent*	*Roundoff*	*Größenordnung*
`float`	*23+1*	*8*	$2^{-24} = 5.96 \times 10^{-8}$	$10^{\pm 38}$
`double`	*52+1*	*11*	$2^{-53} = 1.11 \times 10^{-16}$	$10^{\pm 308}$

festgelegt. Das eigentlich interessante am Standard ist aber die Festlegung des Rundungsverhaltens und die Existenz und Verarbeitung von `NaN`*s*[9]*.*
Der Standard IEEE 854 läßt übrigens als Basis 2 *und* 10 *zu!*

Jetzt sehen wir uns an, wie eine kleine Änderung der Akkumulatorengröße das Verhalten der Subtraktion beeinflussen kann.

Satz 2.11 *Der relative Fehler bei der Subtraktion mit* m*–stelligem Akkumulator kann* $B - 1$ *sein.*

Beweis: Wir setzen $\rho = B - 1$ und betrachten die beiden Zahlen

$$\begin{aligned} x &= .10\cdots 0 \times B^0 = B^{-1} \\ y &= .\rho\cdots\rho \times B^{-1} = B^{-1}\left(1 - B^{-m}\right), \end{aligned}$$

also ist

$$x - y = B^{-1}\left(1 - 1 + B^{-m}\right) = B^{-m-1}.$$

Das berechnete Ergebnis hingegen ist

$$\begin{array}{rl} & .10\cdots 00 \\ - & .0\rho\cdots\rho\rho \\ \hline & .00\cdots 01 \end{array} \quad \rightarrow \quad .10\cdots 0 \times B^{-m+1} = B^{-m}.$$

Der relative Fehler ist somit

$$e_r = \frac{B^{-m} - B^{-m-1}}{B^{-m-1}} = B - 1.$$

□

Satz 2.12 *(Guard Digit) Der relative Fehler bei der Subtraktion mit* $(m + 1)$*–stelligem Akkumulator ist höchstens* $2u$*.*

Beweis: Der kritische Fall ist die Subtraktion zweier Zahlen x, y gleichen Vorzeichens. Nehmen wir an, daß $x, y > 0$ und daß $x > y$. Letzteres ist keine Einschränkung da $x \ominus y = \ominus(y \ominus x)$ und da Vorzeichenumkehr *exakt* durchgeführt werden kann (Vorzeichenbit!). Außerdem können wir die Zahlen so skalieren, daß,

$$x = .x_1\cdots x_m \times B^0, \qquad \text{also} \qquad y = .y_1\cdots y_m \times B^{-k} = .0\cdots 0\hat{y}_{k+1}\cdots\hat{y}_{k+m}$$

[9] `Not a Number`, Ergebnisse von unzulässigen Operationen wie Wurzeln aus negativen Zahlen

für ein $k \in \mathbb{N}_0$ und $x_1, \hat{y}_{k+1} \neq 0$.

Im Fall $k \geq m+1$ haben die ersten $m+1$ Stellen von y, die bei der Rechnung verwendet werden, den Wert 0 und daher ist $x \ominus y = x$. Der Fehler dabei ist $y < B^{-m-1}$ und da $x \geq B^{-1}$, ist $e_r < B^{-m} = \frac{2}{B}\,\frac{1}{2}\,B^{1-m} \leq u$.

Sei also $k \leq m$ und

$$\hat{y} = .0 \cdots 0 \hat{y}_{k+1} \cdots \hat{y}_{m+1}$$

die "Akkumulatorversion" von y (also auf $m+1$ Stellen abgeschnitten). Ist $k = 0$, dann wird die Subtraktion sogar in einem m–stelligen Akkumulator exakt durchgeführt (kein Shift!), ist $k = 1$, so sorgt die "Guard Digit" dafür, daß die Subtraktion im Akkumulator *exakt* ausgeführt wird und Fehler können also nur durch Runden auftreten. Diese sind aber, nach Proposition 2.6 höchstens u. Die folgende Tabelle zeigt, was passiert:

	$.x_1$	x_2	$\cdots$	x_m	0	$0 \ldots 0$
$k=0$	$.y_1$	y_2	$\cdots$	y_m	0	$0 \ldots 0$
$k=1$	.0	y_1	$\cdots$	y_{m-1}	y_m	$0 \ldots 0$

Sei also $k \geq 2$. Dann ist

$$y - \hat{y} = \sum_{j=m+2}^{m+k} \hat{y}_j B^{-j} \leq (B-1) \sum_{j=m+2}^{m+k} B^{-j} = \frac{B-1}{B^{m+2}} \sum_{j=0}^{k-2} B^{-j} = \frac{B-1}{B^{m+2}}\,\frac{1-B^{1-k}}{1-B^{-1}}. \quad (2.1)$$

Nun ist der berechnete Wert $x \ominus y$ gerade

$$x \ominus y = \mathrm{rd}_m\,(x - \hat{y}) = x - \hat{y} - \delta,$$

wobei[10] $|\delta| \leq \frac{B}{2} B^{-m-1} \leq \frac{1}{2} B^{-m}$. Der relative Fehler bei der Subtraktion ist damit

$$e_r = \frac{|(x-y) - (x - \hat{y} - \delta)|}{|x-y|} = \frac{|\hat{y} - y + \delta|}{|x-y|}. \quad (2.2)$$

Wir schreiben $z = x - y = .z_1 \cdots z_{m+k} \times B^0$ (exakte Darstellung!) und unterscheiden drei Fälle:

1. $x - y \geq B^{-1}$ (also $z_1 \neq 0$): dann liefert (2.2) zusammen mit (2.1) die Abschätzung

$$\begin{aligned} e_r &\leq B\,|\hat{y} - y + \delta| \leq B\,(|\hat{y} - y| + |\delta|) \leq B\left(\frac{B-1}{B^{m+2}}\,\frac{1-B^{1-k}}{1-B^{-1}} + \frac{B^{-m}}{2}\right) \\ &= B^{1-m}\left(\frac{B-1}{B^2}\,\frac{1-B^{1-k}}{1-B^{-1}} + \frac{1}{2}\right) = B^{1-m}\left(\frac{B-1}{B-1}\,\frac{1-B^{1-k}}{B} + \frac{1}{2}\right) \\ &= B^{1-m}\left(\frac{1}{2} + \frac{1}{B} - B^{-k}\right) \leq \underbrace{\left(1 + \frac{2}{B}\right)}_{\leq 2}\,\underbrace{\frac{1}{2} B^{1-m}}_{=u} \leq 2u. \end{aligned}$$

[10] Achtung: δ ist hier der *absolute* Fehler bei Rundung auf m Stellen, nicht der relative Fehler wie in Proposition 2.6 betrachtet! Daher auch der um einen Faktor B kleinere Wert.

2. $x - \hat{y} < B^{-1}$: das Ergebnis aus dem Akkumulator muß beim Normalisieren um mindestens eine Stelle nach links geschoben werden und daher passiert beim Runden nichts mehr, also ist $\delta = 0$. Der *kleinstmögliche* Wert für $x - y$ ist nun

$$.10\cdots0 - .\underbrace{0\cdots0}_{k}\underbrace{\rho\cdots\rho}_{m} = .0\underbrace{\rho\cdots\rho}_{k-1}\underbrace{0\cdots0}_{m-1}1 > B^{-2}(B-1)\sum_{j=0}^{k-1}B^{-j},$$

also

$$|x - y| > \frac{B-1}{B^2}\frac{1-B^{-k}}{1-B^{-1}}. \tag{2.3}$$

Setzen wir (2.1), (2.3) und $\delta = 0$ in (2.2) ein, so erhalten wir

$$e_r \leq \frac{B-1}{B^{m+2}}\frac{1-B^{1-k}}{1-B^{-1}} \cdot \frac{B^2}{B-1}\frac{1-B^{-1}}{1-B^{-k}} = B^{-m}\frac{1-B^{1-k}}{1-B^{-k}} < B^{-m} \leq \frac{1}{2}B^{1-m} = u.$$

3. $x - y < B^{-1}$ und $x - \hat{y} \geq B^{-1}$: dies tritt genau dann auf, wenn $x - \hat{y} = B^{-1} = .10\cdots0$ und es ist wieder $\delta = 0$. Damit können wir aber wieder die Abschätzung aus Fall 2 verwenden.

□

Übung 2.5 Beweisen Sie Satz 2.12 für den Fall, daß x und y unterschiedliches Vorzeichen haben. Zeigen Sie, daß diese Aussage sogar *ohne* Verwendung einer "guard digit" gültig bleibt.

Definition 2.13 *Das* Standardmodell *der Gleitkommaarithmetik setzt voraus, daß*

$$x \odot y = (x \cdot y)(1+\delta), \qquad |\delta| \leq \hat{u}, \quad \cdot = +, -, \times, /, \tag{2.4}$$

wobei $\hat{u} = 2u$. Dieses Standardmodell ist mit Hilfe von Guard Digits immer zu realisieren.

Bemerkung 2.14 *(Standardmodell der Gleitkommaarithmetik)*

1. *Das Standardmodell spiegelt die Annahme wieder, daß die jeweilige Rechenoperation* exakt *ausgeführt und dann das Ergebnis gerundet wird. Wir haben aber gesehen, daß, bis auf den Faktor 2 diese Annahme auch mit endlicher Arithmetik realisiert werden kann.*

2. *Das Standardmodell hat noch eine interessante Konsequenz: ist*[11] *$x \cdot y \neq 0$ und ist $\hat{u} < 1$, dann ist auch der berechnete Wert, $x \odot y$, von Null verschieden. Grund: wäre $x \odot y = 0$, dann wäre der relative Fehler der Berechnung*

$$\frac{|x \odot y - x \cdot y|}{|x \cdot y|} = \frac{|x \cdot y|}{|x \cdot y|} = 1 > \hat{u}.$$

Übung 2.6 Zeigen Sie, daß Multiplikation und Division das Standardmodell erfüllen.

Wir wollen im weiteren immer annehmen, daß wir eine Gleitkommaarithmetik zur Verfügung haben, die dem Standardmodell (2.4) genügt.

[11] Eine Arithmetik ohne diese Eigenschaft macht ja auch wenig Sinn

2.2 Fortpflanzung von Rundungsfehlern

Im Normalfall wird bei numerischen Berechnungen nicht nur eine Operation ausgeführt, sondern mehrere hintereinandergeschaltet. Dabei kann die Reihenfolge, in der die Operationen ausgeführt werden, einen *dramatischen* Einfluß auf das Ergebnis haben.

Beispiel 2.15 *Wir wollen für $x, y \in \mathbb{F}$ den Wert $x^2 - y^2$ berechnen.*

1. *Berechnung in der Reihenfolge $(x \otimes x) \ominus (y \otimes y)$. Also ist*

$$\begin{aligned}(x \otimes x) \ominus (y \otimes y) &= x^2\,(1+\varepsilon_1) \ominus y^2\,(1+\varepsilon_2) = \left(x^2(1+\varepsilon_1) - y^2(1+\varepsilon_2)\right)(1+\varepsilon_3) \\ &= \left(x^2 - y^2\right)(1+\varepsilon_3) + \left(\varepsilon_1\, x^2 - \varepsilon_2 y^2\right)(1+\varepsilon_3)\,,\end{aligned}$$

   was zu der pessimistischen Abschätzung

$$\frac{|(x \otimes x) \ominus (y \otimes y) - (x^2 - y^2)|}{|x^2 - y^2|} \leq \hat{u}\left(1 + (1+\hat{u})\frac{x^2+y^2}{|x^2-y^2|}\right) \tag{2.5}$$

   führt, die beliebig schlecht werden kann, wenn $|x| \sim |y|$ ist.

2. *Berechnung in der Reihenfolge $(x \oplus y) \otimes (x \ominus y)$. Dann ist*

$$\begin{aligned}(x \oplus y) \otimes (x \ominus y) &= (x+y)\,(1+\varepsilon_1) \otimes (x-y)\,(1+\varepsilon_2) \\ &= \left((x+y)\,(1+\varepsilon_1) \times (x-y)\,(1+\varepsilon_2)\right)(1+\varepsilon_3) \\ &= \left(x^2 - y^2\right)(1+\varepsilon_1)\,(1+\varepsilon_2)\,(1+\varepsilon_3)\,,\end{aligned}$$

   was die wesentliche bessere Fehlerschranke

$$\frac{|(x \oplus y) \otimes (x \ominus y) - (x^2 - y^2)|}{|x^2 - y^2|} \leq 3\hat{u} + 3\hat{u}^2 + \hat{u}^3 \sim 3\hat{u} \tag{2.6}$$

   ergibt.

Nun muß (2.5) an sich noch nichts schlimmes bedeuten: wir haben lediglich eine *obere* Abschätzung, die sich schlecht verhält. Doch leider ist die Realität auch nicht besser!

Beispiel 2.16 *Wir betrachten*

$$B = 10, \quad m = 3, \qquad x = .334 \quad \textit{und} \quad y = .333.$$

Dann ist $x^2 = .112$, $y^2 = .111$, also hat $x^2 - y^2$ die normalisierte Darstellung $.100 \times 10^{-2}$. Das korrekte Ergebnis hingegen wäre $.667 \times 10^{-3}$, der relative Fehler ist ~ 0.499, also $100u$! Dies ist schlimm genug, allerdings übertreibt Formel (2.5) den relativen Fehler mit einem Wert ~ 333 schon etwas.

```
%%  SkProd1.m (Numerik 1)
%%  ----------------------------------------------------
%%  Berechnung des Skalarprodukts
%%  Eingabe:
%%    x,y  Vektoren

function sp = SkProd( x,y )
  xdim = length( x );
  ydim = length( y );
  sp = 0;

  for i = 1:min( xdim,ydim )
     sp = sp + x(i) * y(i);
  end
%endfunction
```

Programm 2.1 `SkProd1.m`: Berechnung des Skalarprodukts

Als ein weiteres Beispiel von Rundungsfehlerfortpflanzung wollen wir uns das Skalarprodukt zweier Vektoren $x, y \in \mathbb{R}^n$ ansehen. Dies berechnen wir nach der Regel

$$s_0 = 0, \qquad s_j = s_{j-1} \oplus (x_j \otimes y_j), \quad j = 1, \ldots, n,$$

wobei $x^T y = s_n$. Da die Addition von Null unproblematisch ist, ist dann

$$\begin{aligned} s_1 &= x_1 y_1 (1 + \delta_1) \\ s_2 &= (s_1 + x_2 y_2 (1 + \delta_2)) (1 + \varepsilon_2) \\ &= x_1 y_1 (1 + \delta_1) (1 + \varepsilon_2) + x_2 y_2 (1 + \delta_2) (1 + \varepsilon_2), \end{aligned}$$

wobei $|\delta_j|, |\varepsilon_j| \leq \hat{u}$. Der weitere Gang der Dinge ist nun leicht zu erraten.

Lemma 2.17 *Für $j = 2, \ldots, n$ gilt*

$$s_j = \sum_{k=1}^{j} x_k y_k (1 + \delta_k) \prod_{l=k}^{j} (1 + \varepsilon_l), \tag{2.7}$$

wobei $\varepsilon_1 = 0$.

Beweis: Induktion über j. Die Fälle $j = 1, 2$ haben wir ja schon betrachtet. Sei also (2.7) für ein $j \geq 2$ bewiesen. Nun ist

$$s_{j+1} = s_j \oplus (x_{j+1} \otimes y_{j+1}) = (s_j + x_{j+1} y_{j+1} (1 + \delta_{j+1})) (1 + \varepsilon_{j+1})$$

$$
\begin{aligned}
&= s_j\,(1+\varepsilon_{j+1}) + x_{j+1}y_{j+1}\,(1+\delta_{j+1})\,(1+\varepsilon_{j+1}) \\
&= \sum_{k=1}^{j} x_k y_k\,(1+\delta_k)\prod_{l=k}^{j+1}(1+\varepsilon_l) + x_{j+1}y_{j+1}\,(1+\delta_{j+1})\,(1+\varepsilon_{j+1}) \\
&= \sum_{k=1}^{j+1} x_k y_k\,(1+\delta_k)\prod_{l=k}^{j+1}(1+\varepsilon_l)\,.
\end{aligned}
$$

□

Damit erhalten wir den absoluten Fehler

$$
\begin{aligned}
e_a &= \left|s_n - x^T y\right| = \left|\sum_{j=1}^{n} x_j y_j \left((1+\delta_j)\prod_{k=j}^{n}(1+\varepsilon_k) - 1\right)\right| \\
&\le \sum_{j=1}^{n} |x_j y_j|\left((1+\hat{u})^{n-j+2} - 1\right) \\
&= \sum_{j=1}^{n} |x_j y_j|\left((n-j+2)\hat{u} + \binom{n-j+2}{2}\hat{u}^2 + \cdots + \hat{u}^{n-j+2}\right) \\
&\le (n+1)\,\hat{u}\sum_{j=1}^{n}|x_j y_j| + O\left((n\hat{u})^2\right).
\end{aligned}
$$

Definition 2.18 *(Landau–Symbole)*
Sei $\phi : \mathbb{R} \to \mathbb{R}$ stetig in einer Umgebung von 0. Ein Ausdruck $\psi(u)$ ist ein O von $\phi(u)$*, in Zeichen $\psi = O(\phi(u))$, falls*

$$\limsup_{u\to 0}\left|\frac{\psi(u)}{\phi(u)}\right| < \infty. \tag{2.8}$$

Entsprechend schreiben wir $\psi = o(\phi(u))$ falls

$$\lim_{u\to 0}\frac{\psi(u)}{\phi(u)} = 0. \tag{2.9}$$

Da der Rundungsfehler $\hat{u}$ immer als sehr klein (im Vergleich zu n) angenommen wird, also $nu < 1$, werden wir $O\left((n\hat{u})^2\right)$–Terme immer unter den Tisch fallen lassen. Somit haben wir die folgende Aussage über die Stabilität der Berechnung des inneren Produkts.

Satz 2.19 *Der relative Fehler bei der Berechnung des Skalarprodukts $x^T y$, $x, y \in \mathbb{R}$, erfüllt die Ungleichung*

$$e_r = \frac{\left|x^T y - s_n\right|}{|x^T y|} \le n\hat{u}\frac{\sum_{j=1}^{n}|x_j y_j|}{|x^T y|} + O\left((n\hat{u})^2\right). \tag{2.10}$$

Übung 2.7 Erklären Sie, wie man von der Konstante $n+1$ auf n kommt.

Noch ein Wort zur Abschätzung (2.10): ist das Skalarprodukt sehr klein (d.h., die Vektoren sind fast orthogonal), nicht aber die Summe $|x_1y_1| + \cdots + |x_ny_n|$, dann kann die Abschätzung und damit die Genauigkeit des Ergebnisses beliebig schlecht werden. Außerdem muß mindestens einmal Auslöschung auftreten

Wir sehen auch noch eine weitere "Faustregel": die Koeffizienten x_j, y_j sind ja an $n-j+1$ Operationen beteiligt (eine Multiplikation und $n-j$ Additionen) und "sammeln" bei jeder dieser Operationen einen Fehler auf. Da insgesamt n kombinierte Multiplikationen/Additionen auftauchen, ist es nur plausibel, daß der Gesamtfehler die Größenordnung $n\hat{u}$ haben wird. Erstaunlicherweise kann man dieses "Naturgesetz" umgehen, wie wir noch sehen werden.

2.3 Rückwärtsfehler

Wir können Gleichung (2.7) mit $j = n$ noch auf eine andere Weise interpretieren: den berechneten Wert $x \odot y = s_n$ könnte man dadurch erhalten, daß man *exakt* rechnet, aber die Ausgangsdaten stört. Anders gesagt:

$$x \odot y = \hat{x} \cdot \hat{y} = \hat{x}^T \hat{y},$$

wobei

$$\hat{x}_j \hat{y}_j = x_j y_j \left(1 + \delta_j\right) \prod_{k=j}^{n} \left(1 + \varepsilon_k\right).$$

Wir interpretieren Fehler aus der Berechnung also jetzt als Fehler in den Eingabedaten. Diese Idee, die auf Wilkinson [52] zurückgeht, hat die folgenden Vorteile:

- Rundungsfehler und Datenungenauigkeiten (hat da jemand `0.1` eingegeben?) werden gleich behandelt.
- Man kann jetzt auf den reichen Fundus der Störungsrechnung zurückgreifen.
- Die Sensitivität der Rechnung bezüglich individueller Daten wird berücksichtigt (x_1, y_1 sind wesentlich stärker mit Fehlern behaftet als x_n, y_n).

Definition 2.20 *Es sei $f : D \subset \mathbb{R}^n \to \mathbb{R}^m$ eine beliebige Funktion und $\hat{f} : D \to \mathbb{R}^m$ ein numerisches Verfahren dazu. Der* Rückwartsfehler *ρ des Verfahrens ist die kleinste Zahl, so daß es für alle $x \in D \setminus \{0\}$ ein $\hat{x} \in \mathbb{R}^n$ mit der Eigenschaft $\hat{f}(x) = f(\hat{x})$ gibt, so daß die relative Fehlerabschätzung*

$$\frac{\|x - \hat{x}\|}{\|x\|} \leq \rho \tag{2.11}$$

erfüllt ist. Der komponentenweise Rückwärtsfehler *$\rho \in \mathbb{R}^n$ ist analog als kleinste Fehlerschranke in*

$$\frac{|x_j - \hat{x}_j|}{|x_j|} \leq \rho_j, \qquad j = 1, \ldots, n, \tag{2.12}$$

definiert.

Schematisch ist die Idee des Rückwärtsfehlers wie folgt:

$$\begin{array}{ccccc} x & \longrightarrow & \hat{f} & \searrow & \\ \updownarrow & & & & \hat{y} = \hat{f}(x) \\ \hat{x} & \longrightarrow & f & \nearrow & \end{array}$$

Beispiel 2.21 *Nach Lemma 2.17 hat der Rückwärtsfehler für das Skalarprodukt mit vorgegebenem $y \in \mathbb{R}^n$ als Abbildung von $\mathbb{R}^n \to \mathbb{R}^n$ den Wert $\rho = n\hat{u}$, der komponentenweise Rückwärtsfehler hingegen $\rho = (n\hat{u}, n\hat{u}, (n-1)\hat{u}, \ldots, 2\hat{u})$.*

Definition 2.22 *Die* Konditionszahl κ_f *einer Berechnung $f : D \to \mathbb{R}^m$ (in Abhängigkeit von den Eingabewerten $x \in D$) ist eine Schranke für das absolute oder relative Änderungsverhalten*

$$\|f(x+\delta) - f(x)\| \leq \kappa_f(x)\|\delta\|, \qquad \delta \in \mathbb{R}^n,$$

bzw.

$$\|f(x+\delta) - f(x)\| \leq \|f(x)\|\kappa_f(x)\|\delta\|, \qquad \delta \in \mathbb{R}^n,\ f(x) \neq 0,$$

wobei $\|\delta\|$ als sehr klein angenommen wird.

Bemerkung 2.23

1. *Mit der Konditionszahl ergibt sich die "berühmte" Regel*

   $$\text{Fehler} \leq \text{Konditionszahl} \times \text{Rückwärtsfehler}.$$

2. *Diese Aufspaltung ist noch in einer anderen Hinsicht interessant: die Konditionszahl hängt nur von f, also dem Problem, aber* nicht *von $\hat{f}$, dem Verfahren ab und sagt lediglich etwas darüber aus, wie Störungssensitiv das Problem ist. Die Verfahrensfehler werden über die Rückwärtsanalyse in den Rückwärtsfehler gesteckt, wo sie sich mit den Datenfehlern verbünden können.*

3. *Ist $f \in C^2(D)$, dann liefert die Taylorentwicklung*

   $$f(x+\delta) = f(x) + Df^T(x)\,\delta + \frac{1}{2} D^2 f(\theta)[\delta, \delta], \qquad \|\theta - x\| \leq \|\delta\|.$$

   Ist die Bilinearform $D^2 f$ halbwegs vernünftig beschränkt, dann ist der letzte Term ein $O(\|\delta\|^2)$ und die Konditionszahl erfüllt

   $$\kappa_f(x) \sim \|Df\|, \tag{2.13}$$

   wobei $\|\cdot\|$ eine geeignete Norm ist.

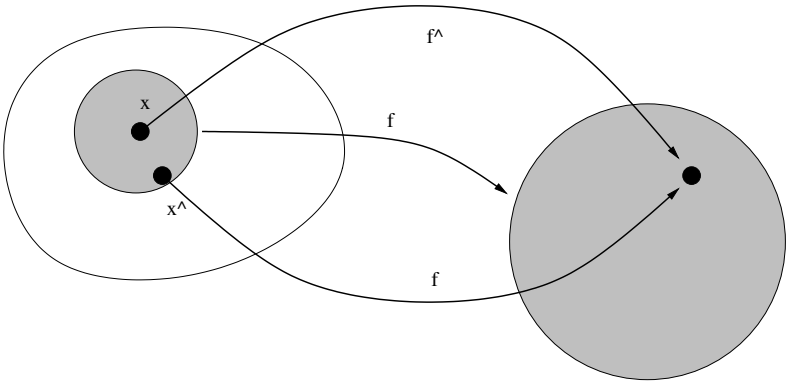


Abbildung 2.1: Rückwärtsfehler und Konditionszahl. Zuerst findet man ein (natürlich möglichst gutes) $\widehat{x}$ so daß $\hat{f}(x) = f(\widehat{x})$, der Rückwärtsfehler ist dann der Radius eines (möglichst kleinen) Kreises um x, der auch $\widehat{x}$ enthält. Dieser Kreis wird nun durch f auf ein anderes Gebiet (hier der Einfachheit halber ebenfalls ein Kreis) abgebildet und das Verhältnis aus dem Radius dieses Gebildes und des Ausgangskreises ist schließlich die Konditionszahl.

Beispiel 2.24 *(Konditionszahl für das Skalarprodukt)*
Halten wir $y \in \mathbb{R}^n$ fest und betrachten die Abbildung

$$f_y : \mathbb{R}^n \to \mathbb{R}, \qquad f_y(x) = x^T y = \sum_{j=1}^{n} x_j y_j.$$

Also ist

$$Df = \left(\frac{\partial f_y}{\partial x_j} : j = 1, \ldots, n \right) = (y_j : j = 1, \ldots, n) = y$$

Auf der anderen Seite ist für das obige Summationsverfahren $\|\delta\| \leq n\hat{u}\|x\|$ (Rückwärtsfehler) und damit haben wir die Abschätzung

$$\left| x^T y - s_n \right| \leq n\hat{u}\|x\| \, \|y\|.$$

Alle Fehlerabschätzungen, die auf Konditionszahlen und Rückwärtsfehlern beruhen, sind *a priori*–Abschätzungen, das heißt, sie sind von den Eingabedaten unabhängig. Einerseits ist das vorteilhaft, weil keine weiteren Voraussetzungen beachtet und eingehalten werden müssen, andererseits haben aber solche "allgemeinen" Aussagen immer den Nachteil, daß sie in vielen Einzelfällen zu einer *Überschatzung* des Fehlers neigen. Bessere, das heißt schärfere Resultate erhält man zumeist durch die sogenannte *runnig error analysis*[12], oder über statistische Methoden, die dann von einer gewissen Verteilung der Eingangsdaten ausgehen und normalerweise Aussagen über mittlere Fehler liefern.

[12]Zumindest der Name sollte einmal gefallen sein.

2.4 Summation

Es ist leicht zu sehen, daß für $x \in \mathbb{R}^n$ der "naive" Summationsalgorithmus

$$s_0 = 0 \qquad s_j = s_{j-1} \oplus x_j, \qquad j = 1, \ldots, n,$$

zur Berechnung von $x^T 1_n = x_1 + \cdots + x_n$, $1_n = (1, \ldots, 1)^T$ als Spezialfall des Skalarprodukts die Fehlerabschätzung

$$\frac{\left|x^T 1_n - s_n\right|}{|x^T 1_n|} \leq n\hat{u} \frac{\sum_{j=1}^n |x_j|}{\left|\sum_{j=1}^n x_j\right|} \tag{2.14}$$

erfüllt. Beim Aufsummieren riesiger Datensätze (z.B. in Statistiken) kann n durchaus groß werden, also ist es möglicherweise nicht sonderlich gut um das Ergebnis der Summation bestellt. Die Frage ist also: "Kann man die Konstante n in der Fehlerschranke verbessern?" Die Antwort ist die "compensated summation"–Formel von Kahan. Der Algorithmus ist allerdings etwas aufwendiger:

$$\begin{array}{rcl} s_1 & = & x_1, \quad c_1 = 0 \\ Y & = & x_j \ominus c_{j-1}, \quad T = s_{j-1} \oplus Y, \quad c_j = (T \ominus s_{j-1}) \ominus Y, \quad s_j = T, \qquad j = 2, \ldots, n. \end{array}$$

Die Idee dieses Verfahrens ist, daß die Folge c_j die Rundungsfehler aufsammelt und immer wieder in die Rechnung steckt und so ein wesentlich besseres Fehlerverhalten bekommt. Den Satz beweisen wir nicht, wer sich für den voll und ganz nichttrivialen Beweis interessiert, findet diesen in [17].

Satz 2.25 *(Rückwärtsfehler für Kahan–Summation)*
Die berechnete Summe s_n der Kahan–Summationsformel erfüllt

$$s_n = \sum_{j=1}^n x_j (1 + \delta_j) + O\left(n\hat{u}^2\right) \sum_{j=1}^n |x_j|, \qquad |\delta_j| \leq 2\hat{u},\ j = 1, \ldots, n. \tag{2.15}$$

Korollar 2.26 *(Vorwärtsfehler für Kahan–Summation)*
Die berechnete Summe s_n der Kahan–Summationsformel hat den relativen Fehler

$$\frac{\left|x^T 1_n - s_n\right|}{|x^T 1_n|} \leq \left(2\hat{u} + O\left(n\hat{u}^2\right)\right) \frac{\sum_{j=1}^n |x_j|}{\left|\sum_{j=1}^n x_j\right|}. \tag{2.16}$$

Übung 2.8 Zeigen Sie: Man kann das Skalarprodukt mit einem Rückwärtsfehler von $3\hat{u} + O(\hat{u}^2)$ berechnen.

```
%%  CompSum1.m
%%  ----------------------------------------------------------
%%  Compensated Summation
%%  Eingabe:
%%    x    Datenvektor

function sum = CompensatedSum( x )
  xdim = length( x );
  S = x(1);
  C = 0;

  for i = 2:xdim
    Y = x(i) - C;
    T = S + Y;
    C = ( T - S ) - Y;
    S = T;
  end

  sum = S;
endfunction
```

Programm 2.2 `CompSum1.m`: Compensated Summation

> $\frac{1}{4}$ *Breite und Länge zusammen sind* 7 *Handbreiten, Länge und Breite zusammen sind* 10 *Handbreiten.*
>
> Babylonisches Gleichungssystem in zwei Unbekannten, Susa, 2. Jahrtausend v. Chr

3 Lineare Gleichungssysteme und numerische Lösungen

Ein lineares Gleichungssystem besteht aus einer Matrix $A \in \mathbb{R}^{m\times n}$ und einem Vektor $b \in \mathbb{R}^m$; das Ziel ist es, einen Vektor $x \in \mathbb{R}^n$ zu finden, so daß

$$Ax = b. \tag{3.1}$$

Dieses Problem ist genau dann für alle *rechten Seiten* $b \in \mathbb{R}^m$ lösbar, wenn A den Rang m hat, also muß, da dieser Rang $\leq \max\{m, n\}$ ist, insbesondere $n \geq m$ gelten. Die Lösung x ist genau dann eindeutig, wenn $m = n$ ist und die (dann) quadratische Matrix A nichtsingulär oder invertierbar ist, das heißt, wenn $\det A \neq 0$ ist.

3.1 Beispiele für lineare Gleichungssysteme

Lineare Gleichungssysteme treten zum Beispiel auf bei

Interpolation: Es sei $\mathscr{F}_n$ ein n–dimensionaler Vektorraum und $f_1, \ldots, f_n$ eine Basis von $\mathscr{F}_n$. Außerdem seien Punkte $x_1, \ldots, x_n$ gegeben. Das *Interpolationsproblem* besteht nun darin, zu vorgegebenen Werten $y_1, \ldots, y_n$ eine Funktion $f = \sum_{j=1}^n a_j f_j$ zu finden, so daß

$$f(x_j) = y_j, \qquad j = 1, \ldots, n.$$

Anders gesagt,

$$\sum_{k=1}^{n} a_k f_k(x_j) = y_j, \qquad j = 1, \ldots, n,$$

oder eben

$$[f_j(x_k) : j, k = 1, \ldots, n]^T a = y. \tag{3.2}$$

Diskretisierung von Operatorgleichungen: Es sei $\mathscr{F}$ ein (möglicherweise unendlichdimensionaler, z.B. $\mathscr{F} = C[0, 1]$) Vektorraum und ein linearer Operator $G : \mathscr{F} \to \mathscr{F}$. Gesucht

wird, zu vorgegebenem $g \in \mathscr{F}$, eine Funktion $f \in \mathscr{F}$ so daß $Gf = g$. Dieses unendlichdimensionale Problem versucht man nun durch endlichdimensionale Probleme anzunähern. Dazu wählt man n–dimensionale Teilräume $\mathscr{F}_n$ und zugehörige Projektionen $P_n : \mathscr{F} \to \mathscr{F}_n$ (*Projektion* heißt, daß $P_n^2 = P_n$, daß also die Abbildung P_n auf $\mathscr{F}_n$ wie die Identität agiert) und sucht eine Funktion $f_n \in \mathscr{F}_n$ so daß

$$P_n G f_n = P_n g.$$

Das heißt, wenn $f_1^n, \ldots, f_n^n$ eine Basis von $\mathscr{F}_n$ ist, dann sucht man Koeffizienten $a_1, \ldots, a_n$, so daß

$$P_n g = P_n G \left(\sum_{k=1}^{n} a_k f_k^n \right) = \sum_{k=1}^{n} (\; P_n \; G f_k^n \;) \, a_k.$$

Da $P_n g \in \mathscr{F}_n$ kann man es als

$$P_n g = \sum_{j=1}^{n} g_j \; f_j^n$$

schreiben und ebenso, für jedes $k = 1, \ldots, n$,

$$\underbrace{P_n \; \underbrace{G f_j^n}_{\in \mathscr{F}}}_{\in \mathscr{F}_n} = \sum_{j=1}^{n} g_{jk} \; f_k^n.$$

Die Matrix $G_n = [g_{jk} : j, k = 1, \ldots, n]$ ist die *Diskretisierung* des Operators G für $\mathscr{F}_n$. Alles in allem erhalten wir also das lineare Gleichungssystem

$$\begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix} = G_n \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}. \tag{3.3}$$

Achtung: dies ist der "triviale" Teil der sogenannten *Galerkin–Methoden*. Das wirkliche Problem besteht darin, *gute* Diskretisierungsräume $\mathscr{F}_n$ zu finden, so daß das Gleichungssystem (3.3) stabil und schnell zu lösen ist. In realistischen Anwendungen sind Größenordnungen wie $n \sim 10^6$ keine Seltenheit.

Bei den Lösungsmethoden für lineare Gleichungssysteme unterscheidet man zwischen *direkten* Methoden (die das Problem in einer endlichen Anzahl von Schritten lösen) und *iterativen* Verfahren (die sich mehr oder weniger langsam an die Lösung herantasten). Außerdem gibt es Verfahren, die die spezielle Struktur bestimmter Matrixtypen ausnutzen, um das Problem schneller oder stabiler zu lösen.

Die meiste Information über das Lösen linearer Gleichungssysteme findet sich in [18, 20], aus denen auch die Kapitel über numerische Lineare Algebra entnommen sind.

Um die Sache einfacher zu machen, werden wir uns im wesentlichen mit quadratischen Matrizen $A \in \mathbb{R}^{n \times n}$ befassen.

3.2 Normen und Konditionszahlen

Um "vernünftige" Aussagen über Fehler und die "numerische Qualität" eines linearen Gleichungssystems machen zu können, benötigen wir Maße für die "Größe" von Vektoren und Matrizen, daher ein bißchen Information über Normen für Vektoren und Matrizen.

Definition 3.1 *Eine Abbildung* $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ *heißt* (Vektor–) Norm*, wenn*

1. $\|x\| \geq 0$ *und* $\|x\| = 0$ *dann und nur dann, wenn* $x = 0$*. (Positivität)*
2. $\|cx\| = |c| \ \|x\|$*,* $c \in \mathbb{R}$*. (Positive Homogenität)*
3. $\|x + y\| \leq \|x\| + \|y\|$*. (Dreiecksungleichung).*

Übung 3.1 Beweisen Sie die weniger bekannte "Dreiecksungleichung nach unten":

$$\|x - y\| \geq \|x\| - \|y\| .$$

Beispiel 3.2 *Klassische Beispiele für Normen sind*

1. *Die* 1–Norm (Manhattan–Norm)
$$\|x\|_1 = \sum_{j=1}^{n} |x_j| .$$
2. *Der* euklidische Abstand
$$\|x\|_2 = \sqrt{\sum_{j=1}^{n} |x_j|^2}.$$
3. *Die* ∞–Norm (Worst–Case–Fehler)
$$\|x\|_\infty = \max_{j=1,\ldots,n} |x_j| .$$
4. *Das sind alles Spezialfälle der sogenannten* p–Normen *(*$1 \leq p \leq \infty$*):*
$$\|x\|_p = \left(\sum_{j=1}^{n} |x_j|^p \right)^{1/p} .$$

Übung 3.2 Es sei $A \in \mathbb{R}^{n\times n}$ eine *symmetrische positiv definite* Matrix, d.h., $A^T = A$ und $x^T Ax > 0$, $x \in \mathbb{R}^n \setminus 0$. Zeigen Sie:

$$\|x\|_A := \sqrt{x^T Ax}, \qquad x \in \mathbb{R}^n,$$

ist eine Norm und plotten Sie für einige A die *Einheitskugeln* $\{x : \|x\|_A = 1\}$.

Analog zu Vektornormen kann man auch Matrixnormen definieren.

Definition 3.3 *Eine Abbildung* $\|\cdot\| : \mathbb{R}^{n\times n} \to \mathbb{R}$ *heißt* (Matrix–) Norm*, wenn*

1. $\|A\| \geq 0$ *und* $\|A\| = 0$ *dann und und nur dann, wenn* $A = 0$*. (Positivität)*
2. $\|cA\| = |c| \ \|A\|$, $c \in \mathbb{R}$*. (Positive Homogenität)*
3. $\|A + B\| \leq \|A\| + \|B\|$*. (Dreiecksungleichung).*

Beispiel 3.4 *Bei Matrixnormen haben wir schon eine viel größere Auswahl:*

1. Vektornormen: *Man faßt die Matrix als einen Vektor in* $\mathbb{R}^{n^2}$ *auf und verwendet eine beliebige Vektornorm, z.B. die* ∞*–Norm*

   $$\|A\|_M = \max_{1\leq j,k\leq n} |a_{jk}|$$

   oder die 2*–Norm*

   $$\|A\|_F = \sqrt{\sum_{j,k=1}^{n} |a_{jk}|^2}.$$

   Die Norm $\|\cdot\|_F$ *heißt* Frobeniusnorm*.*

2. Operatornormen: *zu einer beliebigen Vektornorm* $\|\cdot\|$ *definiert man die Operatornorm*

   $$\|A\| = \max_{x\neq 0} \frac{\|Ax\|}{\|x\|}.$$

3. *Beispiele für die Operatornormen zu den* p*–Normen:*

   Spaltensummennorm:

   $$\|A\|_1 = \max_{k=1,\ldots,n} \sum_{j=1}^{n} |a_{jk}| .$$

   Zeilensummennorm:

   $$\|A\|_\infty = \max_{j=1,\ldots,n} \sum_{k=1}^{n} |a_{jk}| .$$

Übung 3.3 Zeigen Sie: Ist $1/p + 1/q = 1$, dann ist $\|A\|_p = \left\|A^T\right\|_q$.

Definition 3.5 *(Konsistenz und Verträglichkeit)*

1. *Eine Matrixnorm* $\|\cdot\|$ *heißt* konsistent*, falls*

   $$\|AB\| \leq \|A\| \, \|B\| .$$

2. *Eine Matrixnorm* $\|\cdot\|_M$ *und eine Vektornorm* $\|\cdot\|_V$ *heißen* verträglich*, wenn*

$$\|Ax\|_V \leq \|A\|_M \ \|x\|_V .$$

Übung 3.4 Zeigen Sie: Jede Operatornorm ist konsistent und mit der zugehörigen Vektornorm verträglich.

Übung 3.5 Zeigen Sie: Die Norm

$$\|A\| = \max_{j,k=1,\ldots,n} |a_{jk}|$$

ist nicht konsistent.

Übung 3.6 Zeigen Sie: Es gibt keine Matrixnorm $\|\cdot\|$ so daß $\|AB\| = \|A\| \ \|B\|$ für alle $A, B \in \mathbb{R}^{n\times n}$.

Definition 3.6 *Die* Konditionszahl *einer invertierbaren Matrix* $A \in \mathbb{R}^{n\times n}$ *bezüglich der Norm* $\|\cdot\|$ *ist definiert als*

$$\kappa(A) = \left\|A^{-1}\right\| \ \|A\| .$$

Im Falle der p*–Normen schreiben wir* $\kappa_p(A)$*.*

Als nächstes zeigen wir, daß die so definierte Konditionszahl tatsächlich das erfüllt, was wir von einer Konditionszahl erwarten.

Satz 3.7 *Für Matrixnormen* $\|\cdot\|$*, die von einer Vektornorm* $\|\cdot\|$ *abgeleitet sind, gilt*

$$\kappa(A) = \lim_{\varepsilon\to 0} \sup_{\|E\|\leq\varepsilon\|A\|} \frac{\left\|(A+E)^{-1} - A^{-1}\right\|}{\varepsilon \left\|A^{-1}\right\|}. \tag{3.4}$$

Für den Beweis brauchen wir erst eine Hilfsaussage.

Lemma 3.8 *Zu beliebigen Vektoren* $x, y \in \mathbb{R}^n$ *mit* $\|x\| = \|y\| = 1$ *gibt es eine Matrix* $A \in \mathbb{R}^{n\times n}$ *so daß* $y = Ax$ *und* $\|A\| = 1$*.*

Beweis: Wir bezeichnen den *dualen Vektor* zu x mit x', d.h.,

$$x^T x' = 1 \qquad \text{und} \qquad \max_{\|z\|=1} \left|z^T x'\right| = 1.$$

Solch ein Vektor existiert immer (Satz von Hahn–Banach). Dann setzen wir lediglich $A = yx'^T$ und es ist $Ax = y\left(x'^T x\right) = y$ und

$$\|A\| = \max_{\|z\|=1} \|Az\| = \left\|y\left(z^T x'\right)\right\| \leq \|y\| = 1,$$

und da $\|Ax\| = \|y\| = 1$, ist $\|A\| = 1$. □

Beweis von Satz 3.7: Aus der (formalen[13]!) Entwicklung

$$(A+E)^{-1} = \left(\left(I + EA^{-1}\right)A\right)^{-1} = A^{-1}\left(I + EA^{-1}\right)^{-1} = A^{-1}\sum_{j=0}^{\infty}(-1)^j\left(EA^{-1}\right)^j$$

erhalten wir mit $\|E\| \leq \varepsilon\|A\|$, daß

$$(A+E)^{-1} - A^{-1} = -A^{-1}EA^{-1} + O\left(\varepsilon^2\right).$$

Daß diese Vorgehensweise für hinreichend kleine ε korrekt ist, wird Lemma 3.11 zeigen. Wir müssen nun nachweisen, daß

$$\sup_{\|E\|\leq 1}\left\|A^{-1}EA^{-1}\right\| = \left\|A^{-1}\right\|^2, \tag{3.5}$$

denn dann ist

$$\sup_{\|E\|\leq\varepsilon\|A\|}\frac{\left\|(A+E)^{-1} - A^{-1}\right\|}{\varepsilon\left\|A^{-1}\right\|} \leq \frac{\|A\|\ \|A^{-1}\|^2 + O\left(\varepsilon^2\right)}{\varepsilon\left\|A^{-1}\right\|} = \|A\|\left\|A^{-1}\right\| + O\left(\varepsilon\right),$$

woraus (3.4) folgt. Nun ist "$\leq$" in (3.5) einfach:

$$\left\|A^{-1}EA^{-1}\right\| \leq \left\|A^{-1}\right\|\underbrace{\|E\|}_{\leq 1}\left\|A^{-1}\right\| \leq \left\|A^{-1}\right\|^2.$$

Für die Gleichheit wählen wir $y \in \mathbb{R}^n$, $\|y\| = 1$ so, daß

$$\left\|A^{-1}\right\| = \left\|A^{-1}y\right\|$$

(was geht, weil wir eine von einer Vektornorm induzierte Matrixnorm haben) und setzen außerdem

$$x = \frac{A^{-1}y}{\|A^{-1}\|}.$$

Dann ist $\|x\| = \|y\| = 1$ und

$$\left\|A^{-1}EA^{-1}\right\| = \max_{\|z\|=1}\left\|A^{-1}EA^{-1}z\right\| \geq \left\|A^{-1}EA^{-1}y\right\| = \left\|A^{-1}Ex\right\|\ \left\|A^{-1}\right\| \tag{3.6}$$

Nach Lemma 3.8 gibt es eine Matrix $E \in \mathbb{R}^{n\times n}$, so daß $\|E\| = 1$ und $Ex = y$ und somit ist

$$\left\|A^{-1}Ex\right\| = \left\|A^{-1}y\right\| = \left\|A^{-1}\right\|. \tag{3.7}$$

Setzt man (3.7) in (3.6) ein, so ergibt sich schließlich (3.5). □

[13] Das heißt, wir betrachten Reihen ohne uns um deren Konvergenz zu kümmern – also das, wovon man in den Analysis immer geträumt hat.

3.3 Eine allgemeine Fehlerabschätzung

Die (Rückwärts–) Fehleranalyse wird uns später Resultate der Form

$$\widehat{A}\,\widehat{x} = b \tag{3.8}$$

für die berechnete Größe $\widehat{x}$ liefern, wobei die Matrix $E = A - \widehat{A}$ in einem gewissen Sinne "klein" sein wird. Aus diesen Eigenschaften läßt sich dann eine Abschätzung für den relativen Fehler von $\widehat{x}$ herleiten. Zu diesem Zweck bezeichne $\|\cdot\|$ eine beliebige Vektornorm wie auch die zugehörige Operatornorm.

Satz 3.9 *Es sei $A \in \mathbb{R}^{n\times n}$ invertierbar und es gelte*

$$\left\|A^{-1}E\right\| = \rho < 1 \tag{3.9}$$

sowie

$$\|E\| \leq \delta\ \|A\|\,. \tag{3.10}$$

Dann gilt für den relativen Fehler die Abschätzung

$$\frac{\|\widehat{x} - x\|}{\|x\|} \leq \frac{\delta}{1-\rho}\,\kappa(A), \qquad x \in \mathbb{R}^n \setminus \{0\}. \tag{3.11}$$

Bemerkung 3.10 *Wir können die beiden Bedingungen (3.9) und (3.10) zu einer "griffigen" (aber etwas schärferen) Forderung kombinieren, nämlich daß*

$$\kappa(A) < \delta^{-1} \tag{3.12}$$

sein soll. In der Tat, wenn man den "Extremfall" $\delta := \|E\|/\|A\|$ annimmt[14]*, dann ist auch*

$$\rho = \left\|A^{-1}E\right\| \leq \left\|A^{-1}\right\|\|E\| = \frac{\kappa(A)}{\|A\|}\ \|E\| < \delta\ \underbrace{\frac{\|E\|}{\|A\|}}_{=\delta} = 1.$$

Man kann aber (3.12) auch so auffassen, daß das Problem nicht schlechter konditioniert sein darf als die Genauigkeit, mit der $\widehat{A}$ berechnet werden kann, denn ansonsten kann man (heuristisch) davon ausgehen, daß das Ergebnis der Rechnung unsinnig ist. Diese Eigenschaft wird übrigens in Programmen wie `Octave` *oder* `Matlab` *tatsächlich überprüft*[15] *und das Programm liefert eine Warnung, wenn sie nicht erfüllt ist.*

Um Satz 3.9 zu beweisen, erst einmal ein paar Betrachtungen über $\widehat{A}$.

[14] Was für $A \neq 0$ ja kein Problem ist.

[15] Allerdings wird dort **nicht** die "exakte" Konditionszahl bestimmt, was eine $O\left(n^3\right)$–Geschichte wäre, sondern "nur" eine Näherung, die mit einem Aufwand von $O\left(n^2\right)$ bestimmt werden kann. Details findet man in [2].

Lemma 3.11 *Es sei $A \in \mathbb{R}^{n\times n}$ invertierbar. Unter der Voraussetzung (3.9) ist auch die Matrix $\widehat{A} = A - E$ invertierbar und es gilt*

$$\left\|\widehat{A}^{-1}\right\| \leq \frac{\|A^{-1}\|}{1-\rho}. \tag{3.13}$$

Beweis: Wir betrachten

$$\widehat{A} = A - E = A\left(I - A^{-1}E\right)$$

und stellen fest, daß für $N \in \mathbb{N}_0$

$$\left(I - A^{-1}E\right)\left(\sum_{j=0}^{N}\left(A^{-1}E\right)^j\right) = \sum_{j=0}^{N}\left(A^{-1}E\right)^j - \sum_{j=1}^{N+1}\left(A^{-1}E\right)^j = I - \left(A^{-1}E\right)^{N+1}. \tag{3.14}$$

Da

$$\left\|\left(A^{-1}E\right)^{N+1}\right\| \leq \left\|A^{-1}E\right\|^{N+1} \leq \rho^{N+1}$$

konvergiert die rechte Seite von (3.14) gegen I für $N \to \infty$ und damit ist

$$\left(I - A^{-1}E\right)^{-1} = \sum_{j=0}^{\infty}\left(A^{-1}E\right)^j, \tag{3.15}$$

weil die Reihe auf der rechten Seite absolut konvergiert:

$$\left\|\left(I - A^{-1}E\right)^{-1}\right\| = \left\|\sum_{j=0}^{\infty}\left(A^{-1}E\right)^j\right\| \leq \sum_{j=0}^{\infty}\rho^j = \frac{1}{1-\rho}.$$

Also ist

$$\left\|\widehat{A}^{-1}\right\| \leq \left\|A^{-1}\right\| \ \left\|\left(I - A^{-1}E\right)^{-1}\right\| \leq \frac{\|A^{-1}\|}{1-\rho}.$$

□

Beweis von Satz 3.9: Es ist

$$\widehat{x} - x = \widehat{A}^{-1}\left(b - \widehat{A}x\right) = \widehat{A}^{-1}\left(b - (A-E)x\right) = \widehat{A}^{-1}\left(b - Ax + Ex\right) = \widehat{A}^{-1}Ex,$$

also, mittels (3.10) und (3.13)

$$\|\widehat{x} - x\| \leq \left\|\widehat{A}^{-1}\right\| \|E\| \|x\| \leq \frac{\delta}{1-\rho}\left\|A^{-1}\right\| \|A\| \ \|x\|,$$

woraus (3.11) sofort folgt. □

Bemerkung 3.12 *Die implizite Voraussetzung in Satz 3.9, nämlich, daß die Matrixnorm die von der Vektornorm induzierte Operatornorm*[16] *ist, läßt sich etwas abschwächen: Lemma 3.11 setzt lediglich voraus, daß die Matrixnorm konsistent ist, im Beweis von Satz 3.9 verwenden wir, daß die Vektor– und die Matrixnorm verträglich sind. Dies ist aber ohnehin eine natürliche Minimalbedingung, um Vektornormen über eine Matrixnorm beschreiben zu können.*

[16] Man muß schon Mathematiker sein, um das Wort "Norm" dreimal in einem Halbsatz unterzubringen

Vergebens predigt Salomo
Die Leute machens doch nicht so.

Wilhelm Busch

4 Direkte Methoden für lineare Gleichungssysteme

Die "klassische" Methode zur Lösung von Gleichungssystemen ist die *Gauß–Elimination*, die auf der sogenannten LU*–Zerlegung* basiert, bei der das Lösen eines Gleichungssystems auf das Lösen *einfacherer* Gleichungssysteme zurückgeführt wird.

4.1 Dreiecksmatrizen

Wir beginnen mit Gleichungssystemen von besonders einfacher Form, die man mehr oder weniger sofort lösen kann.

Definition 4.1 *Eine Matrix* $A = [a_{jk} : j,k = 1,\ldots,n] \in \mathbb{R}^{n\times n}$ *heißt* untere *(bzw.* obere*)* Dreiecksmatrix*, wenn* $a_{jk} = 0$ *für* $j < k$ *(bzw.* $j > k$*).*

Übung 4.1 Zeigen Sie: das Produkt zweier unterer (oberer) Dreiecksmatrizen ist wieder eine untere (obere) Dreiecksmatrix.

Dreiecksmatrizen definieren besonders einfach zu lösende lineare Gleichungssysteme, bei denen man die Variablen schrittweise eliminieren kann. Ist beispielsweise

$$L = [\ell_{jk} : j,k = 1,\ldots,n]$$

eine untere Dreiecksmatrix, dann lautet das zugehörige Gleichungssystem

$$\begin{array}{lllll} \ell_{11}\,x_1 & & & = & b_1,\\ \ell_{21}\,x_1 & +\ell_{22}\,x_2 & & = & b_2,\\ \vdots & \vdots & \ddots & & \vdots\\ \ell_{n1}\,x_1 & +\ell_{n2}\,x_2 & \ldots\ +\ell_{nn}\,x_n & = & b_n, \end{array}$$

was sofort

$$\begin{aligned} x_1 &= b_1/\ell_{11},\\ x_2 &= (b_2 - \ell_{21}x_1)/\ell_{22},\\ &\vdots \quad \vdots\\ x_n &= (b_n - \ell_{n1}x_1 - \cdots - \ell_{n,n-1}x_{n-1})/\ell_{nn}, \end{aligned}$$

```
%%  VorElim.m (Numerik 1)
%%  ----------------------------------------------------
%%  Vorwaertselimination, ueberschreibt b
%%  Eingabe:
%%    L    untere Dreiecksmatrix
%%    b    rechte Seite

function x = VorElim( L,b )
  n = length( b );

  for j = 1:n
    %% Inneres Produkt!
    b(j) = ( b(j) - L( j,1:j-1 ) * b( 1:j-1 ) ) / L(j,j);
  end

  x = b;
endfunction
```

Programm 4.1 `VorElim.m`: Vorwärtselimination nach (4.1).

ergibt. Die Formel

$$x_j = \left(b_j - \sum_{k=1}^{j-1} \ell_{jk}\, x_k \right) / \ell_{jj}, \qquad j = 1, \ldots, n, \tag{4.1}$$

bezeichnet man als *Vorwärtselimination*, das Gegenstück für obere Dreiecksmatrizen $U = [u_{jk} : j, k = 1, \ldots, n]$, nämlich

$$x_j = \left(b_j - \sum_{k=j+1}^{n} u_{jk}\, x_k \right) / u_{jj}, \qquad j = n, \ldots, 1, \tag{4.2}$$

bezeichnet man als *Rücksubstitution*.

Zuerst ein paar Überlegungen zum Aufwand des Verfahrens: im j–ten Schritt von (4.1) werden $j-1$ Multiplikationen, $j-1$ Subtraktionen und schließlich eine Division ausgeführt, also insgesamt $2j-1$ Fließkommaoperationen, oder kurz *flops* [17]. Insgesamt sind das also

$$\sum_{j=1}^{n} (2j-1) = 2\sum_{j=1}^{n} j - n = n(n+1) - n = n^2$$

flops und genau dasselbe natürlich auch für die Rücksubstitution.

[17] Für *flo*ating point *op*erations

```
%%  RueckSubs.m (Numerik 1)
%%  ------------------------------------------------------
%%  Ruecksubstitution, ueberschreibt b
%%  Eingabe:
%%    U    obere Dreiecksmatrix
%%    b    rechte Seite

function x = RueckSubs( U,b )
  n = length( b );

  b(n) = b(n) / U(n,n);
  for j = n-1 : -1 : 1
    b(j) = ( b(j) - U( j,j+1:n ) * b( j+1:n ) ) / U(j,j);
  end

  x = b;
endfunction
```

Programm 4.2 `RueckSubs.m`: Rücksubstitution nach (4.2).

Bemerkung 4.2 *Das Zählen von flops zur Messung der Komplexität numerischer Verfahren stammt aus der "Ur- und Frühgeschichte" des Computers, als Programmierer noch echte Programmierer und Fließkommaoperationen aufwendig waren. Andere Aufgaben, wie beispielsweise Indizierung und Speicherorganisation waren im Vergleich dazu vernachlässigbar. Dank der modernen Fließkommakoprozessoren ist diese Annahme jedoch* nicht mehr *richtig und daher ist das Zählen von flops auch nur noch von eingeschränktem Wert.*

4.2 Naive Gauß–Elimination

Nachdem also Dreiecksmatrizen sehr einfach zu behandeln sind, ist es sicherlich eine vernünftige Strategie, die Lösung eines allgemeinen Gleichungssystems auf die Lösung von Dreieckssystemen zurückzuführen. Genauer gesagt besteht das Ziel darin, die Matrix A in

$$A = LU$$

zu zerlegen und dann die beiden **einfacheren** Systeme

$$Ly = b \qquad \text{und} \qquad Ux = y$$

durch Vorwärtselimination und Rücksubstitution zu lösen.

Bemerkung 4.3 *Die Berechnung einer LU–Zerlegung hat noch einen weiteren Vorteil: ist das Gleichungssystem $Ax = b$ für* mehrere *rechte Seiten b zu lösen, so muß man die Zerlegung nur einmal durchführen und braucht dann nur noch Dreieckssysteme zu lösen.*

Bevor wir das enstprechende Resultat formulieren können, erst mal ein bißchen Terminologie.

Definition 4.4 *Für eine Matrix $A \in \mathbb{R}^{n\times n}$ bezeichnen wir mit A_m, $m = 1,\ldots,n$, die m–*te Hauptuntermatrix

$$A_m := [a_{jk} : j,k = 1,\ldots,m] \in \mathbb{R}^{m\times m}.$$

*Die Zahl $\det A_m$ bezeichnet man als m–*te Hauptminore *von A.*

Satz 4.5 *Für $A \in \mathbb{R}^{n\times n}$ seien alle Hauptminoren ungleich Null, d.h. $\det A_m \neq 0$, $m = 1,\ldots,n$. Dann gibt es eine untere Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ und eine obere Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$ so daß*

$$A = LU \qquad \textit{und} \qquad \ell_{jj} = 1, \quad j = 1,\ldots,n. \tag{4.3}$$

Bemerkung 4.6 *Die LU–Zerlegung liefert uns außerdem ein wesentlich effizienteres und stabileres Verfahren, um die Determinante einer Matrix zu berechnen, denn in diesem Fall ist ja*

$$\det A = \det(LU) = \underbrace{\det L}_{=1} \det U = \prod_{j=1}^{n} u_{jj}.$$

Korollar 4.7 *Eine Matrix $A \in \mathbb{R}^{n\times n}$ hat genau dann von Null verschiedenen Hauptminoren $\det A_m$, $m = 1,\ldots,n$, wenn es eine untere Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ und eine obere Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$ mit $\ell_{jj} = 1$, $j = 1,\ldots,n$, und $u_{jj} \neq 0$, $j = 1,\ldots,n$, gibt, so daß $A = LU$.*

Beweis: Hat A von Null verschiedene Hauptminoren, so liefern uns Satz 4.5 und

$$0 \neq \det A = \det L \ \det U = \underbrace{\prod_{j=1}^{n} \ell_{jj}}_{=1} \prod_{j=1}^{n} u_{jj} = \prod_{j=1}^{n} u_{jj}, \tag{4.4}$$

die gewünschten Eigenschaften. Für die Umkehrung bemerken wir, daß $A_m = L_m U_m$ (siehe (4.5) für Details), also

$$\det A_m = \prod_{j=1}^{m} u_{jj} \neq 0, \qquad m = 1,\ldots,n.$$

□

Wir werden uns *zwei* Beweise für Satz 4.5 ansehen: der erste ist etwas "direkter" und führt zur Doolittle–Methode, der zweite ist auf den ersten Blick etwas "abstrakter" und wird zur Gauß–Elimination führen, die wir später auf beliebige Matrizen erweitern können.

Beweis von Satz 4.5: Die Zerlegung $A = LU$ ist equivalent zu

$$a_{jk} = \sum_{r=1}^{n} \ell_{jr} u_{rk}, \qquad j,k = 1,\ldots,n,$$

und da $\ell_{jr} = 0$ falls $r > j$ und $u_{rk} = 0$ falls $r > k$, heißt das, daß

$$a_{jk} = \sum_{r=1}^{\min\{j,k\}} \ell_{jr} u_{rk}, \qquad j,k = 1,\dots,n. \tag{4.5}$$

Aus dieser Indentität bauen wir sukzessive die Spalten von L und die Zeilen von U auf; dazu nehmen zuerst einmal an, daß wir für $m = 1,\dots,n$ die ersten $m-1$ Spalten von L und die ersten $m-1$ Zeilen von U, also die Matrizen

$$\widehat{L}_{m-1} = [\ell_{jk} : j = 1,\dots,n,\ k = 1,\dots,m-1] \in \mathbb{R}^{n\times m-1}$$

und

$$\widehat{U}_{m-1} = [u_{jk} : j = 1,\dots,m-1,\ k = 1,\dots,n] \in \mathbb{R}^{m-1\times n}$$

kennen und daß

$$\widehat{L}_{m-1}\widehat{U}_{m-1} = \begin{bmatrix} a_{11} & \dots & a_{1,m-1} & a_{1m} & \dots & a_{1n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m-1,1} & \dots & a_{m-1,m-1} & a_{m-1,m} & \dots & a_{m-1,n} \\ a_{m1} & \dots & a_{m,m-1} & * & \dots & * \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{n,m-1} & * & \dots & * \end{bmatrix}. \tag{4.6}$$

(für $m = 1$ ist dies keine Bedingung!). Nun setzen wir $\ell_{mm} = 1$ und betrachten (4.5) für $j = m$ und $k \geq m$, also

$$a_{mk} = \sum_{r=1}^{m} \ell_{mr} u_{rk} = \sum_{r=1}^{m-1} \ell_{mr} u_{rk} + \underbrace{\ell_{mm}}_{=1} u_{mk}$$

Das lösen wir nach u_{mk} auf und erhalten

$$u_{mk} = a_{mk} - \sum_{r=1}^{m-1} \ell_{mr} u_{rk}, \qquad k = m,\dots,n. \tag{4.7}$$

Wäre nun

$$u_{mm} = a_{mm} - \sum_{r=1}^{m-1} \ell_{mr} u_{rm} = 0,$$

also

$$a_{mm} = \sum_{r=1}^{m-1} \ell_{mr} u_{rm} = \left(\widehat{L}_{m-1}\widehat{U}_{m-1}\right)_{mm},$$

dann wäre, zusammen mit (4.6), $A_m = \left(\widehat{L}_{m-1}\widehat{U}_{m-1}\right)_m$, aber da diese beiden Matrizen jeweils nur Rang $m-1$ haben, könnte A_m nicht invertierbar sein. Also, der langen Rede kurzer Sinn, muß $u_{mm} \neq 0$ sein.

```
%%  Doolittle.m (Numerik I)
%%  ------------------------------------------------------
%%  LU-Zerlegung a la Doolittle
%%  Eingabe:
%%    A    Matrix

function [ L, U ] = Doolittle( A )
  n = length( A );
  L = zeros( n );
  U = zeros( n );

  for m = 1:n
    L( m,m ) = 1;
    for k = m:n
      U( m,k ) = A( m,k ) - L( m,1:m-1 ) * U( 1:m-1,k );
    end
    for j = m+1:n
      L( j,m ) = A( j,m ) - L( j,1:m-1 ) * U( 1:m-1,m );
      L( j,m ) = L( j,m ) / U( m,m );
    end
  end
% endfunction
```

Programm 4.3 `Doolittle.m`: LU–Faktorisierung nach der Methode von Doolittle, (4.7) und (4.8).

Damit können wir den Spieß umdrehen und uns (4.5) für $j > m$ und $k = m$ ansehen, was

$$a_{jm} = \sum_{r=1}^{m} \ell_{jr} u_{rm} = \sum_{r=1}^{m-1} \ell_{jr} u_{rm} + \ell_{jm} u_{mm}$$

und damit

$$\ell_{jm} = \left(a_{jm} - \sum_{r=1}^{m-1} \ell_{jr} u_{rm} \right) / u_{mm}, \qquad j = m+1, \ldots, n, \tag{4.8}$$

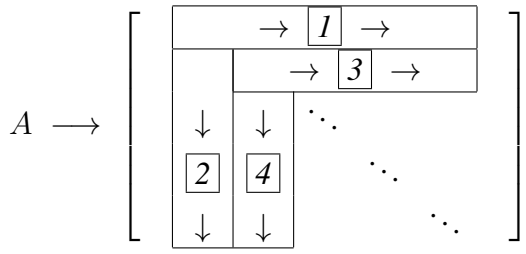
liefert. Außerdem ist das Ganze so gebaut, daß jetzt (4.6) auch mit m anstelle von $m-1$ gilt. □

Das Doolittle–Verfahren ist in `Doolittle.m` realisiert.

Bemerkung 4.8 *Das Doolittle–Verfahren verwendet zur Berechnung eines Eintrags von L oder von U gerade den entsprechenden Eintrag von A sowie vorher berechnete Einträge von L und U. Das erlaubt es, den Eintrag a_{jk} mit*

$$\begin{cases} u_{jk} & j \le k, \\ \ell_{jk} & j > k \end{cases}$$

zu überschreiben. Dabei wird die Matrix gemäß folgendem Schema aufgebaut:



Beispiel 4.9 *Sehen wir uns doch einmal die Funktionsweise des Doolittle–Verfahrens anhand der einfachen Matrix*

$$A = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

an. Um unsere LU–Zerlegung zu bestimmen, müssen wir also eine 3×3–Matrix auffüllen. Wir beginnen mit der ersten Zeile, deren Einträge u_{11}, u_{12} und u_{13} sich nach (4.7) als

$$u_{1k} = a_{1k} - \underbrace{\sum_{r=1}^{0} \ell_{1r} u_{rk}}_{=0} = a_{1k}, \qquad k = 1, 2, 3,$$

bestimmen. Also erhalten wir die erste Zeile unserer "Ergebnismatrix" LU[18] als

$$LU = \begin{bmatrix} \mathbf{2} & \mathbf{1} & \mathbf{0} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}.$$

Jetzt also an die erste Spalte. Nach (4.8) erhalten wir, daß

$$\ell_{j1} = \Big(a_{j1} - \underbrace{\sum_{r=1}^{0} \ell_{jr} u_{r1}}_{=0} \Big) / u_{11} = \frac{a_{j1}}{u_{11}}, \qquad j = 2, 3,$$

und somit

$$LU = \begin{bmatrix} 2 & 1 & 0 \\ \mathbf{\frac{1}{2}} & \cdot & \cdot \\ \mathbf{0} & \cdot & \cdot \end{bmatrix}.$$

Und nun wieder eine Zeile, bei der uns zur Abwechslung (4.7) die Regeln vorgibt:

$$u_{2k} = a_{2k} - \sum_{r=1}^{1} \ell_{2r} u_{rk}, \qquad \Longrightarrow \qquad \begin{array}{rcl} u_{22} & = & a_{22} - \ell_{21}\, u_{12} = -\frac{1}{2}, \\ u_{23} & = & a_{23} - \ell_{21}\, u_{12} = a_{23} = 1. \end{array}$$

[18] Im "oberen" Dreieck dieser Matrix, die Diagonale eingeschlossen, findet sich U, im "unteren" Dreieck die Marix L. Da deren Diagonalemente ℓ_{jj} ja sowieso alle den Wert 1 haben, brauchen wir für sie keinen Speicherplatz zu vergeuden.

Tragen wir das ein, so erhalten wir

$$LU = \begin{bmatrix} 2 & 1 & 0 \\ \frac{1}{2} & -\mathbf{\frac{1}{2}} & \mathbf{1} \\ 0 & \cdot & \cdot \end{bmatrix}.$$

Nun also zur zweiten Spalte, die, nun wieder vermittels (4.8), den Wert

$$\ell_{32} = \left(a_{32} - \sum_{r=1}^{1} \ell_{3r}\, u_{r2} \right) / u_{22} = \left(a_{32} - \ell_{31}\, u_{12} \right) / u_{22} = (1-0) \,/\, \left(-\frac{1}{2} \right) = -2$$

erhält, ergo

$$LU = \begin{bmatrix} 2 & 1 & 0 \\ \frac{1}{2} & -\frac{1}{2} & 1 \\ 0 & \mathbf{-2} & \cdot \end{bmatrix}.$$

Wie sich die Zerlegung schließlich zu

$$LU = \begin{bmatrix} 2 & 1 & 0 \\ \frac{1}{2} & -\frac{1}{2} & 1 \\ 0 & -2 & \mathbf{2} \end{bmatrix}$$

vervollständigen läßt, ist jetzt kein Rätsel mehr. Tatsächlich kann man das Doolittle–Verfahren auch ohne jedewede Form mathematischer Kenntnisse durchführen, solange man sich nur an das einfache Schema aus Abb. 4.1 hält.

Das Programm 4.4 `Doolittle2.m` realisiert das Doolittle–Verfahren mit sofortigem Überschreiben von A, also auf speicherplatzsparende Art.

Bestimmen wir schließlich noch den Rechenaufwand des Doolittle–Verfahrens: um, für ein festes m den Wert u_{mk} zu berechnen brauchen wir $m-1$ Multiplikationen und ebensoviele Subtraktionen, also insgesamt $2m-2$ Operationen. Für die Einträge ℓ_{jm} sind, wegen der Division jeweils $2m-1$ Operationen nötig. Für jedes m sind außerdem die $n-m+1$ Einträge von U und $n-m$ Einträge von L zu berechnen (da $\ell_{jj}=1$), also brauchen wir für ein festes m insgesamt

$$(n-m+1)(2m-2) + (n-m)(2m-1) = 4nm - 4m^2 + 5m - 3n - 2$$

Summiert man nun $m = 1, \ldots, n$, dann ergibt sich

$$\frac{2}{3}\, n^3 - \frac{1}{2}\, n^2 + \frac{11}{6}\, n - 2$$

für den Rechenaufwand des Doolittle–Verfahrens.

Bemerkung 4.10 *Generell beträgt der Aufwand bei Faktorisierungsverfahren $O\left(n^3\right)$, das wird bei der Gauß–Elimination auch so sein. Dies ist eine Größenordnung mehr als bei der Vorwärtselimination und der Rücksubstitution.*

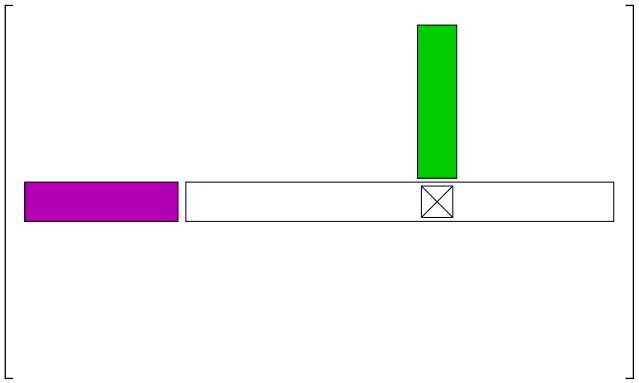


Abbildung 4.1: Schematische Darstellung des Doolittle–Verfahrens. Bei der Bestimmung der m–ten Zeile wird der neue Eintrag dadurch bestimmt, daß man vom "Originalwert" der Matrix A das innere Produkt der beiden (farbigen) $(m-1)$–Vektoren am Anfang der Zeile und der Spalte, in denen sich der Eintrag befindet, abzieht. Bei der Bestimmung einer Spalte geht das ganz genauso, nur wird zusätzlich noch durch den Wert des Diagonalelements dividiert.

Nun zur Gauß–Elimination.

Definition 4.11 *Eine* Gauß–Transformation *oder* Gauß–Matrix *ist eine Matrix der Form*

$$M_k = I - ye_k^T, \qquad y = \begin{bmatrix} 0 \\ \widehat{y} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ y_{k+1} \\ \vdots \\ y_n \end{bmatrix}, \quad \widehat{y} \in \mathbb{R}^{n-k}, \qquad k = 1, \ldots, n-1.$$

Bemerkung 4.12 *(Gauß–Transformationen)*

1. *Gauß–Transformationen sind untere Dreiecksmatrizen der Form*

$$M_k = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -y_{k+1} & 1 & \\ & & \vdots & & \ddots \\ & & -y_n & & 1 \end{bmatrix}.$$

2. *Die Inverse der Gauß–Transformation $I - ye_k^T$ ist $I + ye_k^T$:*

$$\left(I - ye_k^T\right)\left(I + ye_k^T\right) = I - ye_k^T + ye_k^T - y \underbrace{e_k^T y}_{=y_k=0} e_k^T = I.$$

```
%%  Doolittle2.m (Numerik I)
%%  -------------------------------------------------------
%%  Doolittle-Methode mit Ueberschreiben
%%  Eingabe:
%%    A    Matrix

function LU = Doolittle2( A )
  n = length( A );

  for m = 1:n
    for k = m:n
      A( m,k ) = A( m,k ) - A( m,1:m-1 ) * A( 1:m-1,k );
    end
    for j = m+1:n
      A( j,m ) = A( j,m ) - A( j,1:m-1 ) * A( 1:m-1,m );
      A( j,m ) = A( j,m ) / A( m,m );
    end
  end
  LU = A;
% endfunction
```

Programm 4.4 `Doolittle2.m`: Doolittle–Methode mit Überschreiben von A.

3. Für eine Matrix

$$A = \begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix}$$

mit Zeilenvektoren $a_1^T, \ldots, a_n^T$ ist

$$M_k A = \begin{bmatrix} a_1^T \\ \vdots \\ a_k^T \\ a_{k+1}^T - y_{k+1}\, a_k^T \\ \vdots \\ a_n^T - y_n\, a_k^T \end{bmatrix} = A - y\, a_k^T,$$

das heißt, von den Zeilen a_j^T, $j = k+1, \ldots, n$, wird das y_j–fache der Zeile a_k^T abgezogen.

Beweis von Satz 4.5: Die Strategie besteht darin, Gauß–Transformationen

$$M_j = I - y^{(j)} e_j^T, \qquad j = 1, \ldots, n-1,$$

zu finden, so daß

$$M_{n-1} \cdots M_1 A =: U$$

eine obere Dreiecksmatrix ist, denn dann haben wir sofort die Zerlegung

$$A = \underbrace{M_1^{-1} \cdots M_{n-1}^{-1}}_{=:L}\, U. \tag{4.9}$$

Dazu konstruieren wir induktiv die Gauß–Transformationen $M_1, M_2, \ldots$ so, daß

$$M_m \cdots M_1 A = \begin{bmatrix} U_m & B_m \\ 0 & \tilde{A}_m \end{bmatrix}, \qquad m = 0, \ldots, n-1, \tag{4.10}$$

wobei

$$U_m \in \mathbb{R}^{m \times m}, \quad B_m \in \mathbb{R}^{m \times n-m}, \quad \tilde{A}_m \in \mathbb{R}^{n-m \times n-m} \tag{4.11}$$

und $(U_m)_{jj} \neq 0$, $j = 1, \ldots, m$. Dann ist mit $\tilde{A}_0 = A$ die Bedingung (4.10) für $m = 0$ erfüllt. Nehmen wir also an, wir hätten für ein $m \geq 0$ die Zerlegung (4.10) bestimmt. Da die Hauptminoren die Beziehung

$$A_{m+1} = \left(M_1^{-1} \cdots M_m^{-1}\right)_{m+1} \begin{bmatrix} U_m & B_m \\ 0 & \tilde{A}_m \end{bmatrix}_{m+1}$$

erfüllen, ist also

$$0 \neq \det A_{m+1} = \det \begin{bmatrix} U_m & B_m \\ 0 & \tilde{A}_m \end{bmatrix}_{m+1} = \det \begin{bmatrix} U_m & * \\ 0 & \left(\tilde{A}_m\right)_{11} \end{bmatrix} = \left(\tilde{A}_m\right)_{11} \prod_{j=1}^{m} (U_m)_{jj},$$

und somit muß $\left(\tilde{A}_m\right)_{11} \neq 0$ sein. Damit setzen wir

$$y_j^{(m+1)} = \begin{cases} 0 & j = 1, \ldots, m+1 \\ \frac{\left(\tilde{A}_m\right)_{j-m,1}}{\left(\tilde{A}_m\right)_{11}} & j = m+2, \ldots, n \end{cases}$$

und $M_{m+1} = I - y^{(m+1)} e_{m+1}^T$ und erhalten (4.10) für $m+1$ anstelle von m. Bleibt nur noch zu bemerken, daß $M_n = I$ ist. □

Bemerkung 4.13 *Auch hier erhalten wir wieder aus der Invertierbarkeit der Hauptminoren, daß die Diagonalelemente*

$$u_{jj} = \left(\tilde{A}_{m-1}\right)_{11} \neq 0$$

sein müssen.

Sehen wir uns noch kurz die Matrix L aus (4.9) an.

Lemma 4.14 *Für Gauß–Transformationen $M_j = I - y^{(j)} e_j^T$ gilt*

$$\prod_{j=1}^{m} M_j^{-1} = I + \sum_{j=1}^{m} y^{(j)} e_j^T = \begin{bmatrix} 1 & & & & & \\ y_2^{(1)} & \ddots & & & & \\ \vdots & \ddots & 1 & & & \\ y_{m+1}^{(1)} & \cdots & y_{m+1}^{(m)} & 1 & & \\ \vdots & \ddots & \vdots & & \ddots & \\ y_n^{(1)} & \cdots & y_n^{(m)} & & & 1 \end{bmatrix}. \tag{4.12}$$

Beweis: Induktion über m, $m = 1$ ist klar. Außerdem ist, per Induktionsannahme

$$\prod_{j=1}^{m+1} M_j^{-1} = \left(I + \sum_{j=1}^{m} y^{(j)} e_j^T\right)\left(I + y^{(m+1)} e_{m+1}^T\right) = I + \sum_{j=1}^{m+1} y^{(j)} e_j^T + \sum_{j=1}^{m} y^{(j)} \underbrace{e_j^T y^{(m+1)}}_{=y_j^{(m+1)}=0} e_{m+1}^T.$$

□

Bemerkung 4.15 *(Realisierung der Gauß–Elimination)*

1. *Nach (4.12) erhalten wir die Matrix L einfach dadurch, daß wir die zu den jeweiligen Gauß–Transformationen gehörigen Vektoren $y^{(j)}$ aufsammeln.*
2. *Damit können wir die Gauß–Transformation wieder durch Überschreiben der Werte in A berechnen.*

```
%%  Gauss1.m (Numerik 1)
%%  ----------------------------------------------------
%%  Gauss-Elimination mit Ueberschreiben (ohne Pivot)
%%  Eingabe:
%%    A    Matrix

function LU = Gauss1( A )
  n = length( A );

  for m = 1:n
    for j = m+1:n
      y = A( j,m ) / A( m,m );
      A( j,m ) = y;
      A( j,m+1:n ) = A( j,m+1:n ) - y * A( m,m+1:n );
    end
  end
  LU = A;
% endfunction
```

Programm 4.5 `Gauss1.m`: Gauß–Elimination mit Überschreiben der Ausgangsmatrix.

Beispiel 4.16 (Implementierung der Gauß–Elimination) *Nachdem wir unseren ersten "richtigen" Algorithmus hergeleitet haben, sehen wir uns doch einmal an, wie man sowas in* `Octave` *umsetzt. Es ist immer gut, sich erst einmal interaktiv anzusehen, was man eigentlich macht, also nehmen wir uns mal eine zufällige*[19]*, aber nicht zu große Matrix vor:*

```
octave> A = rand(5)
A =

  0.9225764  0.7304416  0.0053629  0.4718802  0.5500260
  0.5163327  0.3250707  0.9407548  0.1423454  0.7346591
  0.9691272  0.2264440  0.8408027  0.0108411  0.5356029
  0.0816184  0.7742376  0.9738590  0.4147111  0.0869881
  0.5917631  0.8076336  0.5608915  0.5655123  0.9978877
```

So, was passiert jetzt gleich wieder bei der Gauß–Elimination? Richtig, wir ziehen passende Vielfache der ersten Zeile *von allen folgenden Zeilen ab, und die Faktoren sind die Werte der ersten Spalte dividiert durch den Wert "oben links". Nachdem* `A(a:b,c:d)` *ja die Teilmatrix mit den Indexmengen* `(a:b)` *und* `(c:d)` *liefert, können wir uns recht einfach die Zeile und die Spalte holen:*

[19] Das ist natürlich bei jedem Ausprobieren eine andere Matrix, also nicht erschrecken!

```
octave> y = A( 1,1:5 )
y =

  0.9225764  0.7304416  0.0053629  0.4718802  0.5500260

octave> x = A( 1:5,1 ) / A( 1,1 )
x =

  1.000000
  0.559664
  1.050457
  0.088468
  0.641425
```

Man beachte: Zeile und Spalte passen. Die Elimination besteht nun darin, daß wir die Matrix[20] `x * y` *von* `A` *abziehen:*

```
octave> A - x*y
ans =

   0.00000   0.00000   0.00000   0.00000   0.00000
   0.00000  -0.08373   0.93775  -0.12175   0.42683
   0.00000  -0.54085   0.83517  -0.48485  -0.04218
   0.00000   0.70962   0.97338   0.37296   0.03833
   0.00000   0.33911   0.55745   0.26284   0.64509
```

Was ist nun mit unserer ersten Zeile passiert? Nun, wenn wir sie von sich selbst abziehen, dann muß ja Null rauskommen. Das können wir mit `y(1) = 0` *ganz einfach korrigieren, aber warum sollten wir denn eigentlich überhaupt die erste Zeile verändern, denn eigentlich findet die Elimination ja sowieso nur in der* 4×4*–Submatrix "unten rechts" statt – wenn wir uns mit* `x` *nicht blöd angestellt haben,* ***müssen*** *wir in der ersten Spalte ja Nullen bekommen. Der langen Rede kurzer Sinn: Wir operieren nur auf der unteren Matrix und machen es ein bißchen anders.*

```
octave> y = A( 1,2:5 )
y =

  0.7304416  0.0053629  0.4718802  0.5500260

octave> x = A( 2:5,1 ) / A( 1,1 )
x =

  0.559664
  1.050457
```

[20] Spalte mal Zeile gibt eine Matrix!

```
  0.088468
  0.641425
```

eliminieren im 4×4*–Teil*

```
octave> A( 2:5,2:5 ) = A( 2:5,2:5 ) - x*y
A =

   0.9225764   0.7304416   0.0053629   0.4718802   0.5500260
   0.5163327  -0.0837311   0.9377534  -0.1217489   0.4268294
   0.9691272  -0.5408538   0.8351692  -0.4848490  -0.0421760
   0.0816184   0.7096170   0.9733845   0.3729649   0.0383285
   0.5917631   0.3391105   0.5574516   0.2628368   0.6450875
```

und schreiben das "kleine" `x` *in die erste Spalte:*

```
octave> A( 2:5, 1 ) = x
A =

   0.9225764   0.7304416   0.0053629   0.4718802   0.5500260
   0.5596639  -0.0837311   0.9377534  -0.1217489   0.4268294
   1.0504574  -0.5408538   0.8351692  -0.4848490  -0.0421760
   0.0884679   0.7096170   0.9733845   0.3729649   0.0383285
   0.6414245   0.3391105   0.5574516   0.2628368   0.6450875
```

Und das war auch schon der erste Schritt Gauß–Elimination mit Überschreiben! Jetzt müssen wir das nur noch auf die Teilmatrizen anwenden, das heißt, alle vorherigen Schritte wiederholen, aber eben nicht bei 1*, sondern bei* $2, 3, 4$ *anfangen. Das ist dann auch in Algorithmus 4.6 implementiert.*

Bestimmen wir noch schnell den Aufwand der Gauß–Elimination: für festes m kostet die Elimination einer Zeile $2(n - m)$ Operationen und außerdem 1 Operation für die Bestimmung von y. Insgesamt sind $n - m$ Zeilen zu eliminieren, was also einen Gesamtaufwand von $(n-m)(2n-2m+1)$ ergibt. Insgesamt, nach Summation über m, kostet die Gauß–Elimination also

$$\frac{2}{3}n^3 - \frac{1}{2}n^2 - \frac{1}{6}n$$

flops .

4.3 Das fertige Lösungsverfahren

Nun können wir aus den Lösungen der Teilprobleme schließlich ein Verfahren basteln, das die Lösung des Gleichungssystems $Ax = b$ berechnet:

1. Bestimme die Zerlegung
$$A = LU \tag{4.13}$$
mittels Gauß–Elimination oder nach Doolittle.

```
%%  Gauss2.m (Numerik 1)
%%  -----------------------------------------------------
%%  Gauss-Elimination mit Ueberschreiben
%%  (ohne Pivot, aber mit mehr Matlab-Features)
%%  Eingabe:
%%    A    Matrix

function LU = Gauss2( A )
  n = length( A );

  for m = 1:n-1
    a = A( m,m );           % Pivotelement
    y = A( m,m+1:n );       % Zeile
    x = A( m+1:n,m ) / a;   % Spalte (normalisiert)
    A ( m+1:n,m+1:n ) = A( m+1:n,m+1:n ) - x*y;
    A ( m+1:n,m ) = x;
  end
  LU = A;
% endfunction
```

Programm 4.6 `Gauss2.m`: Gauß–Elimination nach Beispiel 4.16.

```
%%  LoesAx-b1.m (Numerik 1)
%%  ------------------------------------------------------
%%  Loesen eines Gleichungssystems (ohne Pivot)
%%  Eingabe:
%%    A    Matrix
%%    b    rechte Seite

function x = LoesAxb1( A,b )
  LU = Gauss1a( A );
  y = VorElim1( LU,b );
  x = RueckSubs( LU,y );
% endfunction
```

Programm 4.7 `LoesAxb1.m`: Lösung eines Gleichungssystems nach (4.13)–(4.15).

2. Bestimme die Lösung y von
$$Ly = b \tag{4.14}$$
durch Vorwärtselimination.

3. Bestimme die Lösung x von
$$Ux = y \tag{4.15}$$
durch Rücksubstitution.

Bemerkung 4.17 *Man kann auch Vorwärtselimination und Rücksubstitution direkt auf die in einer einzelnen Matrix gespeicherte LU–Zerlegung anwenden, wenn man nur die Vorwärtselimiation so modifiziert, daß sie alle Diagonalelemente als 1 annimmt, wie in Programm* `VorElim1.m` *gezeigt.*

4.4 Fehleranalyse

In diesem Kapitel wollen wir uns nun Abschätzungen für den *Rückwärtsfehler* der Gauß–Elimination ansehen, wie man daraus einen relativen Fehler "bastelt", ist ja aus Satz 3.9 bekannt.

Definition 4.18 *Zu einer Matrix $A \in \mathbb{R}^{n\times n}$ definieren wir die "Betragsmatrix" $|A|$ als*
$$|A| = [|a_{jk}| : j,k = 1,\ldots,n]\,.$$
Zu Matrizen $A, B \in \mathbb{R}^{n\times n}$ definieren wir die Halbordnung[21] *"$\leq$" als*
$$A \leq B \qquad \Longleftrightarrow \qquad a_{jk} \leq b_{jk}, \quad j,k = 1,\ldots,n.$$

[21] Zur Erinnerung: Bei einer *totalen* Ordnung sind je zwei Elemente $x \neq y$ vergleichbar, das heißt, es ist entweder $x < y$ oder $x > y$, bei einer *Halb*ordnung muß dies nicht der Fall sein.

```
%%  VorElim1.m (Numerik 1)
%%  -----------------------------------------------------
%%  Vorwaertselimination, ueberschreibt b, Diagonale = 1!!!
%%  Eingabe:
%%    L    untere Dreiecksmatrix
%%    b    rechte Seite

function x = VorElim1( L,b )
  n = length( b );

  for j = 1:n
    b(j) = b(j) - L(j,1:j-1) * b(1:j-1);
  end

  x = b;
endfunction
```

Programm 4.8 `VorElim1.m`: Vorwärtselimination unter der Annahme, daß alle Diagonalemente von L den Wert 1 haben.

Lemma 4.19 *Für $A, B \in \mathbb{R}^{n\times n}$ gilt*

$$|AB| \leq |A|\,|B|.$$

Beweis: Für $1 \leq j,k \leq n$,

$$\left|(AB)_{jk}\right| = \left|\sum_{r=1}^{n} a_{jr}\; b_{rk}\right| \leq \sum_{r=1}^{n} |a_{jr}|\;\; |b_{rk}| = \left(|A|\,|B|\right)_{jk}.$$

□

Bemerkung 4.20 *Wir machen von nun an die* generelle Annahme*, daß $n\hat{u} \ll 1$ ist (sonst ergibt diese Form der numerischen Linearen Algebra sowieso keinen Sinn) und schreiben nur noch $O\left(\hat{u}^2\right)$ anstelle des (korrekteren) $O\left(n\hat{u}^2\right)$*

Wenden wir nun das Verfahren (4.13)–(4.15) an, dann erhalten wir also zuerst eine näherungsweise LU–Zerlegung $\widehat{L}$ und $\widehat{U}$, dann eine Näherungslösung $\widehat{y}$ (die aus $\widehat{L}$, nicht aus L berechnet wird!) und schließlich ein berechnetes $\widehat{x}$ (das aus $\widehat{U}$ und $\widehat{y}$ berechnet wird). Die Rückwärtsfehleraussage für dieses Lösungsverfahren lautet dann wie folgt.

Satz 4.21 *Sei $A \in \mathbb{F}^{n\times n}$ LU–zerlegbar und $\hat{x}$ die mittels Gauß–Elimination, Vorwärtselimination und Rücksubstitution berechnete Lösung von $Ax = b$. Dann gibt es eine Matrix $\widehat{A}$, so daß*

$$\widehat{A}\,\widehat{x} = b \tag{4.16}$$

und

$$\left|\widehat{A} - A\right| \leq n\hat{u}\left(3\,|A| + 5\left|\widehat{L}\right|\,\left|\widehat{U}\right|\right) + O\left(\hat{u}^2\right). \tag{4.17}$$

Bemerkung 4.22 *Die Forderung $A \in \mathbb{F}^{n\times n}$ ist nur marginal. Für beliebiges A bekommen wir noch einen (fast vernachlässigbaren) Fehler E mit*

$$|E| \leq \hat{u}\,|A|$$

dazu.

Bevor wir mit dem Beweis von Satz 4.21 beginnen, sollten wir uns kurz überlegen, was er uns eigentlich sagen will. Immerhin ist es ja das erste[22] Beispiel für eine ernsthafte Bestimmung eines Rückwärtsfehlers. Bei dieser Rückwärtsfehleranalyse im Sinne von Definition 2.20 fassen wir die Lösung x des linearen Gleichungssystems $Ax = b$ als Funktion $x = f(A, b)$ von A und b auf, während das *numerische* Verfahren bestehend aus Gauß–Elimination, Vorwärtselimination und Rücksubstitution uns den *berechneten* Wert $\widehat{x} = \widehat{f}(A, b)$ liefert. Nun schieben wir alle Rundungsfehler auf den ersten Parameter A und bestimmen eine Matrix $\widehat{A}$, so daß $\widehat{x} = f\left(\widehat{A}, b\right)$ ist, und dann schätzen wir ab, inwieweit sich A und $\widehat{A}$ unterscheiden – das ist gerade (4.17). Daß alle Fehler ausschließlich auf A geschoben werden ist zuerst einmal total *willkürlich*[23], zumindest aus der Sicht der Rückwärtsfehleranalyse könnten wir genausogut auch einen Teil der Rundungsfehler als Störung von b interpretieren. Es gibt allerdings einen guten Grund, das nicht zu machen: Die Lösung ist ja $x = A^{-1}b$ und so gesehen können wir unser Problem genausogut als numerische Invertierung von A und folgende exakte Multiplikation von b auffassen und die *Konditionszahl* dieser Operation ist gerade $\kappa(A)$, also auch eine Größe, die wir "kennen".

Bleibt noch die Frage, wie man sowas macht, wie man Rückwärtsfehler eines Verfahrens bestimmt. Die generelle Vorgehensweise ist eigentlich immer dieselbe: Man geht sein Verfahren Schritt für Schritt durch und bestimmt für jeden dieser Schritte die Rundungsfehler beziehungsweise eine möglichst scharfe obere Schranke für diese. Das ist auch schon die Idee des nun folgenden Beweises, alles weitere ist Handwerk und eine sorgfältige Analyse des Verfahrens; ein weiterer Vorteil einer Rundungsfehleranalyse ist es tatsächlich, daß man dabei neben gesicherten Fehleraussagen auch oftmals Schwachstellen oder kritische Punkte des Algorithmus findet.

Jetzt aber endlich zum Beweis selbst. Zuerst einmal sehen wir uns den Fehler bei der Vorwärtselimination und Rücksubstitution an. Dazu wird es opportun sein, auch das *alternative* Modell der Gleitkommaarithmetik zu verwenden, das annimmt, daß

$$x \odot y = \frac{x \cdot y}{1+\delta}, \qquad |\delta| \leq \hat{u}, \quad \cdot = +, -, \times, /. \tag{4.18}$$

Diese Annahme ist (ohne Beweis!) durchaus sinnvoll, denn mit $\hat{u} = 4u$ kann man auf alle Fälle garantieren, daß (2.4) und (4.18) gleichzeitig gelten.

Übung 4.2 Zeigen Sie:

[22] Und glücklicherweise auch einzige.

[23] Aber das ist die Auswahl von Sündenböcken eigentlich immer

1. Zu jedem $0 \neq x \in D(\mathbb{F})$ gibt es ein $0 \neq \hat{x} \in \mathbb{F}$, so daß

$$\hat{x} = x\,(1+\delta)\,, \qquad |\delta| \leq u.$$

2. Zu jedem $\delta \in [-\hat{u}, \hat{u}]$ gibt es ein $\vartheta \in [-\hat{u}, 2\hat{u}]$, so daß

$$1 + \delta = \frac{1}{1+\vartheta}.$$

Jetzt aber zum Rückwärtsfehler der Vorwärtselimination.

Proposition 4.23 *Die durch Vorwärtselimination (Rücksubstitution) berechnete Lösung $\hat{x}$ von $Lx = b$, ($Ux = b$) $L, U \in \mathbb{R}^{n\times n}$, erfüllt $\hat{L}\hat{x} = b$ ($\hat{U}\hat{x} = b$), wobei $\hat{L}$ ($\hat{U}$) ebenfalls eine untere (obere) Dreicksmatrix ist, die*

$$\left|\hat{L} - L\right| \leq n\hat{u}\,|L| + O\left(\hat{u}^2\right), \tag{4.19}$$

beziehungsweise

$$\left|\hat{U} - U\right| \leq n\hat{u}\,|U| + O\left(\hat{u}^2\right), \tag{4.20}$$

erfüllt.

Beweis: Wir bezeichnen mit $\ell^j = [\ell_{jk} : k = 1, \ldots, j-1] \in \mathbb{R}^{j-1}$ den Anfang des j–ten Zeilenvektors von L[24]. Mit der Vorwärtselimination bestimmen wir nun Vektoren $\hat{x}^j \in \mathbb{R}^j$, $j = 1, \ldots, n$, durch

$$\hat{x}^j = \left(b_j \ominus \ell^{j-1} \odot \hat{x}^{j-1}\right) \oslash \ell_{jj}, \qquad x^0 = 0,$$

wobei $\hat{x}^n = \hat{x}$. Nun wissen wir von den Skalarprodukten (Beispiel 2.21) bereits, daß für $x \in \mathbb{F}^{j-1}$

$$\ell^j \odot \hat{x} = \sum_{k=1}^{j-1} \ell_{jk}\hat{x}_k\,(1+\delta_{jk}) = \sum_{k=1}^{j-1} (1+\delta_{jk})\,\ell_{jk}\,\hat{x}_k,$$

wobei $|\delta_{jk}| \leq (j-1)\hat{u}$. Mittels (4.18) bekommen wir somit, daß

$$\hat{x}^j = \frac{b_j - \ell^j \odot \hat{x}^{j-1}}{(1+\eta_j)\,\ell_{jj}}, \qquad |\eta_j| \leq 2\hat{u},$$

da η_j den Fehler einer Subtraktion und einer Division aufnimmt. Wir brauchen also nur noch

$$\hat{\ell}_{jk} = \begin{cases} (1+\delta_{jk})\,\ell_{jk}, & j < k \\ (1+\eta_j)\,\ell_{jj}, & j = k, \end{cases}$$

zu setzen und erhalten sogar, daß

$$\left|\hat{L} - L\right| \leq \max\{2, n-1\}\;\hat{u}\;|L|\; + O\left(\hat{u}^2\right) \tag{4.21}$$

[24] Also das, was *unterhalb* der Diagonalen steht

gilt, also insbesondere (4.19).
Der Beweis für die Rücksubstitution funktioniert natürlich ganz analog. □

Übung 4.3 Geben Sie ein Verfahren an, das die Vorwärtselimination bzw. Rücksubstitution mit einem von n unabhängigen Rückwärtsfehler ausführt.

Bemerkung 4.24 *Gehen wir nur vom Standardmodell (2.4) aus, dann ist (4.21) mit der Konstante* $\max\{4, n-1\}$ *erfüllt, was für* $n \geq 5$ *auch kein Beinbruch ist.*

Jetzt aber zur Fehleranalyse der Gauß–Elimination selbst.

Proposition 4.25 *Sei* $A \in \mathbb{F}^{n\times n}$. *Wenn alle Hauptminoren von* A *invertierbar sind, dann erfüllt die berechnete* LU*–Zerlegung* $\widehat{L}, \widehat{U}$ *die Bedingung*

$$\left|A - \widehat{L}\widehat{U}\right| \leq 3n\hat{u}\left(|A| + \left|\widehat{L}\right|\ \left|\widehat{U}\right|\right) + O\left(\hat{u}^2\right). \tag{4.22}$$

Beweis: Durch Induktion über n. Im Fall $n = 1$ ist $\widehat{U} = A$, $\widehat{L} = 1$, und (4.22) ist trivialerweise richtig (die linke Seite hat Wert 0!).

Sei also (4.22) für ein $n \geq 1$ bewiesen und sei $A \in \mathbb{R}^{n+1\times n+1}$, das wir schreiben als

$$A = \left[\begin{array}{c|c} a_{11} & w^T \\ \hline v & B \end{array}\right], \qquad v, w \in \mathbb{R}^n,\ B \in \mathbb{R}^{n\times n}. \tag{4.23}$$

Nach Annahme, $a_{11} \neq 0$, können wir für den ersten Schritt der Gauß–Elimination den Vektor $\widehat{y}^{(1)} = v \oslash a_{11}$ und die Matrix $\widehat{A}_1 = B \ominus \widehat{y}^{(1)} \otimes w^T$ als Näherung des "eliminierten" $A_1 = B - \widehat{y}^{(1)} \otimes w^T$ berechnen. Dabei ist

$$\left|\widehat{y}^{(1)} - y^{(1)}\right| \leq \hat{u}\left|y^{(1)}\right| = \hat{u}\,\frac{|v|}{|a_{11}|}$$

sowie

$$\left|\widehat{y}^{(1)} \otimes w^T - y^{(1)}w^T\right| \leq 2\hat{u}\left|y^{(1)}w^T\right| + O\left(\hat{u}^2\right),$$

also

$$\left|\widehat{A}_1 - A_1\right| \leq 3\hat{u}\ |A_1| + O\left(\hat{u}^2\right). \tag{4.24}$$

Nach Induktionsannahme ergibt die (numerische!) Zerlegung von $\widehat{A}_1$ dann Matrizen $\widehat{L}_1, \widehat{U}_1$, so daß

$$\left|\widehat{A}_1 - \widehat{L}_1\widehat{U}_1\right| \leq 3n\hat{u}\left(\left|\widehat{A}_1\right| + \left|\widehat{L}_1\right|\ \left|\widehat{U}_1\right|\right) + O\left(\hat{u}^2\right).$$

Wir erhalten dann, daß

$$\widehat{L} = \left[\begin{array}{c|c} 1 & 0 \\ \hline \widehat{y}^{(1)} & \widehat{L}_1 \end{array}\right], \qquad \widehat{U} = \left[\begin{array}{c|c} a_{11} & w^T \\ \hline 0 & \widehat{U}_1 \end{array}\right],$$

wobei die erste Zeile von $\widehat{U}$ wegen der 1 "oben links" in $\widehat{L}$ *exakt* ist. Damit erhalten wir, daß

$$\begin{aligned}
\left|A-\widehat{L}\widehat{U}\right| &= \left|\left[\begin{array}{c|c} a_{11} & w^T \\ \hline v & B \end{array}\right] - \left[\begin{array}{c|c} a_{11} & w^T \\ \hline a_{11}\,\widehat{y}^{(1)} & \widehat{y}^{(1)}w^T+\widehat{L}_1\widehat{U}_1 \end{array}\right]\right| \\
&= \left|\left[\begin{array}{c|c} 0 & 0 \\ \hline v-a_{11}\,\widehat{y}^{(1)} & B-\widehat{y}^{(1)}w^T-\widehat{L}_1\widehat{U}_1 \end{array}\right]\right| = \left|\left[\begin{array}{c|c} 0 & 0 \\ \hline v-a_{11}\,\widehat{y}^{(1)} & A_1-\widehat{L}_1\widehat{U}_1 \end{array}\right]\right| \\
&\le \left|\left[\begin{array}{c|c} 0 & 0 \\ \hline a_{11}\left(\widehat{y}^{(1)}-y^{(1)}\right) & A_1-\widehat{A}_1 \end{array}\right]\right| + \left|\left[\begin{array}{c|c} 0 & 0 \\ \hline v-a_{11}\,y^{(1)} & \widehat{A}_1-\widehat{L}_1\widehat{U}_1 \end{array}\right]\right| \\
&\le \left|\left[\begin{array}{c|c} 0 & 0 \\ \hline a_{11}\,\hat{u}|v|/\left|a_{11}\right| & A_1-\widehat{A}_1 \end{array}\right]\right| + \left|\left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & \widehat{A}_1-\widehat{L}_1\widehat{U}_1 \end{array}\right]\right| \\
&\le 3\hat{u}\left[\begin{array}{c|c} 0 & 0 \\ \hline |v| & |A_1| \end{array}\right] + 3n\hat{u}\left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & |A_1|+\left|\widehat{L}_1\right|\,\left|\widehat{U}_1\right| \end{array}\right] \\
&\le 3(n+1)\hat{u}\left[\begin{array}{c|c} |a_{11}| & \left|w^T\right| \\ \hline |v| & |A_1|+\left|\widehat{L}_1\right|\,\left|\widehat{U}_1\right| \end{array}\right] = 3(n+1)\hat{u}\left(|A|+\left|\widehat{L}\right|\,\left|\widehat{U}\right|\right).
\end{aligned}$$

Und das war's dann auch schon. □

Beweis von Satz 4.21: Aus Proposition 4.23 erhalten wir, daß

$$\left(\widehat{L}+F\right)\hat{y}=b, \qquad \left(\widehat{U}+E\right)\hat{x}=\hat{y},$$

wobei

$$|E|\le n\hat{u}|\widehat{U}|+O\left(\hat{u}^2\right), \qquad |F|\le n\hat{u}|\widehat{L}|+O\left(\hat{u}^2\right),$$

also

$$\left(\widehat{L}\widehat{U}+G\right)\hat{x}=b,$$

wobei

$$|G|=\left|\widehat{L}E+F\widehat{U}+FE\right|\le 2n\hat{u}\left|\widehat{L}\right|\,\left|\widehat{U}\right|+O\left(\hat{u}^2\right).$$

Nach Proposition 4.25 ist damit $\widehat{A}\hat{x}=b$, wobei

$$\begin{aligned}
\left|\widehat{A}-A\right| &= \left|G+\widehat{L}\widehat{U}-A\right| \le |G|+\left|A-\widehat{L}\widehat{U}\right| \\
&\le 2n\hat{u}\left|\widehat{L}\right|\,\left|\widehat{U}\right|+3n\hat{u}\left(|A|+\left|\widehat{L}\right|\,\left|\widehat{U}\right|\right)+O\left(\hat{u}^2\right) \\
&\le n\hat{u}\left(3\,|A|+5\left|\widehat{L}\right|\,\left|\widehat{U}\right|\right)+O\left(\hat{u}^2\right).
\end{aligned}$$

□

4.5 Pivotsuche

Was sagt uns Satz 4.21 nun über die numerische Stabilität der Gauß–Elimination? Nun, Einsetzen in die allgemeine Fehlerabschätzung (3.9) ergibt das folgende Resultat.

Korollar 4.26 *Für eine Vektornorm $\|\cdot\|$ und eine konsistente, mit der Vektornorm verträgliche und "vernünftige"[25] Matrixnorm sei γ so gewählt, daß*

$$\left\| \left|\widehat{L}\right| \left|\widehat{U}\right| \right\| \leq \gamma \|A\| . \tag{4.25}$$

Dann ist

$$\frac{\|\hat{x} - x\|}{\|x\|} \leq \frac{n\hat{u}\,(3+5\gamma)}{1 - n\hat{u}\,(3+5\gamma)\,\kappa(A)}\kappa(A), \qquad x \neq 0, \tag{4.26}$$

vorausgesetzt, daß $n\hat{u}\,(3+5\gamma)\,\kappa(A) < 1$.

Bemerkung 4.27 *Setzt man $B := |A|$, dann ist $|A| \leq |B| \leq |A|$, also, für eine monotone Matrixnorm, $\|A\| \leq \|B\| \leq \|A\|$ und damit ist $\| |A| \| = \|A\|$. Wegen*

$$\left|\widehat{L}\right| \left|\widehat{U}\right| \geq \left|\widehat{L}\widehat{U}\right| = \left|\widehat{A}\right|$$

wird γ in (4.25) einen Wert > 1 haben.

Leider kann der Wert γ sehr schnell sehr groß werden.

Beispiel 4.28 *Für $B = 10$, $m = 3$ und abschneidende Arithmetik wird die Matrix*

$$A = \begin{bmatrix} .100 \times 10^{-2} & .100 \times 10^{1} \\ .100 \times 10^{1} & .200 \times 10^{1} \end{bmatrix}$$

durch Gauß–Elimination zerlegt in

$$\widehat{L} = \begin{bmatrix} .100 \times 10^{1} & .000 \times 10^{0} \\ .100 \times 10^{4} & .100 \times 10^{1} \end{bmatrix}, \qquad \widehat{U} = \begin{bmatrix} .100 \times 10^{-2} & .100 \times 10^{1} \\ .000 \times 10^{0} & -.100 \times 10^{4} \end{bmatrix}.$$

Dann ist

$$\widehat{L}\widehat{U} = \begin{bmatrix} .100 \times 10^{-2} & .100 \times 10^{1} \\ .100 \times 10^{1} & .000 \times 10^{0} \end{bmatrix},$$

aber

$$\left|\widehat{L}\right| \left|\widehat{U}\right| = \begin{bmatrix} .001 & 1 \\ 1 & 2000 \end{bmatrix}$$

Der Eintrag "rechts unten", nämlich 2000, ist übrigens von der Größenordnung $20\ \hat{u}^{-1}$, da $\hat{u} = B^{-2} = 1/100$.

Wo liegt nun das Problem? Normalerweise werden bei der Gauß–Elimination Vielfache von Zeilen von A von "weiter unten liegenden" Zeilen abgezogen. Dieses Vielfache wird dann betraglich sehr groß, wenn der (Diagonal–) Wert, durch den dividiert wird, sehr klein wird. Daher empfiehlt es sich, Strategien zu suchen, die den Divisor nicht zu klein werden lassen.

[25] Der korrekte Begriff ist der einer *monotonen* Matrixnorm, d.h., eine Norm mit der Eigenschaft $|A| \leq |B| \Longrightarrow \|A\| \leq \|B\|$

Definition 4.29 *Das Element a_{11} einer Matrix $A \in \mathbb{R}^{n\times n}$ bezeichnet man als* Pivotelement.

Definition 4.30 *Eine Matrix $P \in \mathbb{R}^{n\times n}$ heißt* Permutationsmatrix, *wenn*

$$P \in \{0,1\}^{n\times n} \qquad \textit{und} \qquad \sum_{r=1}^{n} p_{rk} = \sum_{r=1}^{n} p_{jr} = 1, \quad j,k = 1,\ldots,n. \tag{4.27}$$

Die Vertauschungsmatrix $P[j,k]$, $j,k = 1,\ldots,n$, *ist definiert als*

$$P[j,k] = \begin{bmatrix} 1 & & & & & & & & & \\ & \ddots & & & & & & & & \\ & & 1 & & & & & & & \\ & & & 0 & & & 1 & & & \\ & & & & 1 & & & & & \\ & & & & & \ddots & & & & \\ & & & & & & 1 & & & \\ & & & 1 & & & & 0 & & \\ & & & & & & & & 1 & \\ & & & & & & & & & \ddots \\ & & & & & & & & & & 1 \end{bmatrix} = I - (e_j - e_k)\,(e_j - e_k)^T$$

Wir bezeichnen mit

$$\Pi_n = \{P[j,k] : 1 \le j < k \le n\}$$

die Menge aller Vertauschungsmatrizen. Diese erzeugen dann auch die (multiplikative) Gruppe[26] *der Permutationsmatrizen.*

Übung 4.4 Zeigen Sie:

1. Jede Permutationsmatrix P ist als Produkt von Vertauschungsmatrizen darstellbar und jedes Produkt
$$\prod_{j=1}^{k} P_j, \qquad P_j \in \Pi_n,\ k \in \mathbb{N},$$
ist wieder eine Permutationsmatrix.

2. Die Menge der Permutationsmatrizen bildet eine Gruppe.

Bemerkung 4.31 *Für $P = P[j,k] \in \Pi_n$ und $A \in \mathbb{R}^{n\times n}$ ist PA diejenige Matrix, bei der die* Zeilen *j und k von A vertauscht werden und AP diejenige Matrix, bei der die* Spalten *j und k von A vertauscht werden. Außerdem ist $P^2 = I$ für jedes $P \in \Pi_n$*

Damit haben wir das allgemeine Resultat zur Gauß–Elimination.

[26] Jede Permutation hat eine Inverse und das Produkt zweier Permutationen ist wieder eine Permutation.

Satz 4.32 *Sei $A \in \mathbb{R}^{n\times n}$ invertierbar. Dann gibt es Permutationsmatrix $P \in \mathbb{R}^{n\times n}$, eine untere Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ mit $\ell_{jj} = 1$ und eine obere Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$, so daß*

$$PA = LU. \tag{4.28}$$

Korollar 4.33 *Eine Matrix $A \in \mathbb{R}^{n\times n}$ ist genau dann invertierbar, wenn es eine Permutationsmatrix P (d.h., eine Umordnung der Zeilen) gibt, so daß alle Hauptminoren $(PA)_m$, $m = 1, \ldots, n$, von PA invertierbar sind.*

Beweis: Korollar 4.7 und Satz 4.32. □

Bemerkung 4.34 *(Lösungsverfahren)*

1. *Da $Ax = b$ äquivalent zu $Pb = PAx = LUx$ ist, können wir, sobald die LU–Zerlegung und P bestimmt sind, das gesuchte x wieder über die Gleichungssysteme $Ly = Pb$, $Ux = y$ bestimmen.*

2. *Da Permutationen* exakt *bestimmt werden können, ist die Fehleranalyse von (4.28) exakt dieselbe wie die für die naive Gauß–Elimination.*

3. *Satz 4.32 ergibt als Verfahren die* Gauß–Elimination mit Spaltenpivotsuche*, alternativ könnte man auch Zerlegungen*

$$\begin{array}{lll} \textit{“Zeilenpivot”} & AQ = LU, & Q \in \Pi_n, \\ \textit{“Totalpivot”} & PAQ = LU, & P, Q \in \Pi_n \end{array} \tag{4.29}$$

   untersuchen. Die zugehörigen Lösungsmethoden sind

$$\begin{array}{lll} Ly = b, & Uz = y, & x = Qz, \\ Ly = Pb, & Uz = y, & x = Qz. \end{array}$$

4. *Aus* $\det P = 1$ *und (4.28) folgt, daß*

$$0 \neq \det A = \underbrace{(\det P)^{-1}}_{=1} \underbrace{\det L}_{=1} \det U = \prod_{j=1}^{s} u_{jj},$$

   also $u_{jj} \neq 0$, $j = 1, \ldots, n$.

Beweis von Satz 4.32: Wir gehen im wesentlichen wie im “naiven” Fall (zweiter Beweis von Satz 4.5) vor und konstruieren wie dort Gauß–Transformationen M_j und Permutationsmatrizen $P_j \in \Pi_n$, $j = 1, \ldots, n$, so daß

$$A^{(m)} := M_m P_m \cdots M_1 P_1 A = \begin{bmatrix} U_m & B_m \\ 0 & \widetilde{A}_m \end{bmatrix}, \qquad m = 0, \ldots, n, \tag{4.30}$$

wobei U_m, B_m und $\widetilde{A}_m$ wie in (4.11) sind.[27] Nehmen wir also an, wir hätten für ein $m \geq 0$ so eine Zerlegung (für $m = 0$ ist dies keine Bedingung, also wieder mal ein trivialer Induktionsanfang), dann betrachten wir die erste Spalte von $\widetilde{A}_m$ und wählen k so, daß

$$|\widetilde{a}_{k1}| \geq |\widetilde{a}_{j1}|, \qquad j = 1, \ldots, n-m. \tag{4.31}$$

Ist $\widetilde{a}_{k1} = 0$, dann hat $A^{(m)}$ die Form

$$A^{(m)} = \left[\begin{array}{c|cc} U_m & b & B \\ \hline 0 & 0 & c^T \\ 0 & 0 & \widetilde{A} \end{array}\right]$$

und damit bestenfalls den Rang $n-1$, ist aber auf keinen Fall invertierbar, was aber nach der Voraussetzung ausgeschlossen ist. Ansonsten setzen wir $P_{m+1} = P[m+1, m+k]$, das heißt

$$PA^{(m)} = \begin{bmatrix} U_m & B_m \\ 0 & C_m \end{bmatrix}, \qquad C := \underbrace{P[1,k]\widetilde{A}_m}_{\in \Pi_n},$$

sowie

$$y_j^{(m+1)} = \begin{cases} 0, & j = 1, \ldots, m+1 \\ \frac{c_{j-m,1}}{c_{11}}, & j = m+2, \ldots, n, \end{cases}$$

und $M_{m+1} = I - y^{(m+1)} e_{m+1}^T$. Da in der Matrix C das betragsgrößte Element der ersten Spalte an c_{11} stand, ist insbesondere

$$\left| y_j^{(m+1)} \right| \leq 1. \tag{4.32}$$

Außerdem ist, genau wie in der naiven Gauß–Elimination nun

$$A^{(m+1)} = M_{m+1} P_{m+1} A^{(m)} = \begin{bmatrix} U_{m+1} & B_{m+1} \\ 0 & \widetilde{A}_{m+1} \end{bmatrix},$$

so daß die Zerlegung aus (4.30) auf den Fall $m+1$ erweitert ist. Nach n Schritten haben wir dann also, daß

$$M_n P_n \cdots M_1 P_1 A = U, \qquad M_n = P_n = I, \tag{4.33}$$

und damit ist $A = (M_{n-1} P_{n-1} \cdots M_1 P_1)^{-1} U$. Was wir noch zeigen müssen, ist, daß es eine untere Dreiecksmatrix $L \in \mathbb{R}^{n \times n}$ und eine Permutation P gibt, so daß

$$(M_{n-1} P_{n-1} \cdots M_1 P_1)^{-1} = P_1 M_1^{-1} \cdots P_{n-1} M_{n-1}^{-1} = PL, \tag{4.34}$$

denn dann ist

$$P^{-1} A = P^{-1} P_1 M_1^{-1} \cdots P_{n-1} M_{n-1}^{-1} U = P^{-1} PLU = LU \tag{4.35}$$

und natürlich ist P^{-1} auch wieder eine Permutation.

[27] Dies ist genau das Gegenstück zu (4.10).

Bleibt also nur noch (4.34); das funktioniert, weil wir Gauß–Transformationen und Permutationen in einem gewissen Sinne vertauschen können, solange nur die Indizes zusammenpassen. Dazu schreiben wir $M_j = M_j\left(y^{(j)}\right)$ und verwenden die Indentität

$$M_j\left(y^{(j)}\right)\, P\,[k,\ell] = P\,[k,\ell]\; M_j\left(P\,[k,\ell]\, y^{(j)}\right), \qquad 1 \le j < k < \ell \le n, \tag{4.36}$$

aus der (4.34) unmittelbar folgt. Aber auch (4.36) ist nicht schwer zu beweisen: unter Verwendung der Kurzschreibweise $y = y^{(j)}$, $M = M_j\left(y^{(j)}\right)$ und $P = P[k,\ell]$ erhalten wir, daß

$$\begin{aligned} PMP &= P\left(I - y\, e_j^T\right) P = \underbrace{P^2}_{=I} - (Py)\; e_j^T \underbrace{\left(I - (e_k - e_\ell)\,(e_k - e_\ell)^T\right)}_{=P} \\ &= \underbrace{I - (Py)\; e_j^T}_{=M(Py)} + (Py)\,(\underbrace{e_j^T e_k}_{=0} - \underbrace{e_j^T e_\ell}_{=0})\; (e_k - e_\ell)^T = M\,(Py)\,, \end{aligned}$$

und das war's dann auch, wenn man bedenkt, daß $P = P^{-1}$ und damit $MP = PM(Py)$ ist. □

Beispiel 4.35 *Wir betrachten*[28] *die Matrix*

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

und führen die Gauß–Elimination mit Spaltenpivotsuche von Hand durch. Das Pivotelement in der ersten Spalte ist natürlich a_{13}, also zerlegen wir

$$P[1,3]\; A = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 3 \\ 4 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 2 & 1 \\ 2 & 1 & 3 \\ 1 & 0 & 0 \end{bmatrix}$$

in

$$M_1 P[1,3]\; A = \begin{bmatrix} 1 & & \\ -\frac{1}{2} & 1 & \\ -\frac{1}{4} & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 2 & 1 \\ 2 & 1 & 3 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 4 & 2 & 1 \\ 0 & 0 & \frac{5}{2} \\ 0 & -\frac{1}{2} & -\frac{1}{4} \end{bmatrix}.$$

Jetzt müssen wir sogar vertauschen und erhalten

$$P[2,3] M_1 P[1,3]\; A = \begin{bmatrix} 4 & 2 & 1 \\ 0 & -\frac{1}{2} & -\frac{1}{4} \\ 0 & 0 & \frac{5}{2} \end{bmatrix},$$

was schon eine obere Dreiecksmatrix ist. Also ist

$$A = \underbrace{P[1,3]^{-1}}_{=P[1,3]} M_1^{-1} \underbrace{P[2,3]^{-1}}_{=P[2,3]} \begin{bmatrix} 4 & 2 & 1 \\ 0 & -\frac{1}{2} & -\frac{1}{4} \\ 0 & 0 & \frac{5}{2} \end{bmatrix},$$

[28] Es wurde ein Beispiel "mit Zahlen" geünscht.

wobei

$$\begin{aligned} P[1,3]M_1^{-1}P[2,3] &= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & & \\ \frac{1}{2} & 1 & \\ \frac{1}{4} & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & & \\ \frac{1}{4} & 1 & \\ \frac{1}{2} & 0 & 1 \end{bmatrix} \\ &= P[1,3]\,P[2,3] \begin{bmatrix} 1 & & \\ \frac{1}{4} & 1 & \\ \frac{1}{2} & 0 & 1 \end{bmatrix}. \end{aligned}$$

Die entscheidende Aktion der "neuen" Gauß–Elimination ist die *Spaltenpivotsuche* in (4.31): man bestimmt eine Permutation $P_{m+1} = P[m+1,k]$, $k \geq m+1$, so daß

$$\left|\left(P_{m+1}A^{(m)}\right)_{m+1,m+1}\right| \geq \left|\left(P_{m+1}A^{(m)}\right)_{j,m+1}\right|, \qquad j = m+2,\ldots,n. \tag{4.37}$$

Analog bestimmt man bei der *Zeilenpivotsuche* eine Permutation $Q_{m+1} = P[m+1,k']$, $k' \geq m+1$, so daß

$$\left|\left(A^{(m)}Q_{m+1}\right)_{m+1,m+1}\right| \geq \left|\left(A^{(m)}Q_{m+1}\right)_{m+1,j}\right|, \qquad j = m+2,\ldots,n, \tag{4.38}$$

und bei der *Totalpivotsuche* sogar *zwei* Permutationen $P_{m+1} = P[m+1,k]$, $Q_{m+1} = P[m+1,k']$, $k,k' \geq m+1$, so daß

$$\left|\left(P_{m+1}A^{(m)}Q_{m+1}\right)_{m+1,m+1}\right| \geq \left|\left(P_{m+1}A^{(m)}Q_{m+1}\right)_{j,j'}\right|, \qquad j,j' = m+2,\ldots,n. \tag{4.39}$$

Die Gauß–Elimination *mit* Spaltennpivotsuche ist in `GaussSP.m` realisiert, das endgültige Lösungsverfahren unter Berücksichtigung der Permutation in `LoesAxb2.m`.

Der einzige Unterschied zur Gauß–Elimination ohne Pivotsuche besteht darin, daß bei der Pivotsuche Zeilen vertauscht werden müssen und die entsprechende Permutationsmatrix zu speichern ist. Diese beiden Operationen sind natürlich *exakt* durchführbar. Daher haben wir sofort die folgende Fehleraussage.

Korollar 4.36 *Sei $A \in \mathbb{F}^{n\times n}$ und $\hat{x}$ die mittels Gauß–Elimination mit Spaltenpivotsuche, Vorwärtselimination und Rücksubstitution berechnete Lösung von $Ax = b$. Dann gibt es eine Matrix $\widehat{A}$, so daß*

$$\widehat{A}\,\hat{x} = b \tag{4.40}$$

und

$$\left|\widehat{A} - A\right| \leq n\hat{u}\left(3\,|A| + 5\left|\widehat{L}\right|\,\left|\widehat{U}\right|\right) + O\left(\hat{u}^2\right). \tag{4.41}$$

Bemerkung 4.37 *(Beliebte Mißverständnisse)*

```
%%  GaussSP.m (Numerik 1)
%%  --------------------------------------------------------
%%  Gauss-Elimination mit Ueberschreiben, Spaltenpivotsuche
%%  und Matlab-Features
%%  Eingabe:
%%    A    Matrix

function [ LU, p ] = GaussSP( A )
  n = length( A );
  p = [1:n];

  for m = 1:n
                                % Suche Pivot

    piv = abs( A(m,m) );
    pivj = m;
    for j = m+1:n
      if abs( A(j,m) ) > piv
        piv = abs( A(j,m) );
        pivj = j;
      end
    end
                                % Vertausche

    j = p( m ); p( m ) = p( pivj ); p( pivj ) = j;
    x = A( m,: ); A( m,: ) = A( pivj,: ); A( pivj,: ) = x;

                                % Eliminiere

    w = A( m,m+1:n );
    for j = m+1:n
      A( j,m ) = A( j,m ) / A( m,m );
      A( j,m+1:n ) = A( j,m+1:n ) - A( j,m ) * w;
    end
  end
  LU = A;
% endfunction
```

Programm 4.9 `GaussSP.m`: Gauß–Elimination mit Spaltenpivotsuche. Der Vektor `p` enthält die Permutation: `k = p(j)` bedeutet, daß $Pe_k = e_j$.

```
%%  LoesAxb2.m (Numerik 1)
%%  ------------------------------------------------------
%%  Loesen eines Gleichungssystems (mit Pivot)
%%  Eingabe:
%%    A    Matrix
%%    b    rechte Seite

function x = LoesAxb2( A,b )
  [ LU,p ] = GaussSP( A );

  for j = 1:length(A)
    bb( j ) = b( p(j) );
  end

  y = VorElim1( LU,bb );
  x = RueckSubs( LU,y );
% endfunction
```

Programm 4.10 `LoesAxb2.m`: Lösung eines Gleichungssystems vermittels Gauß–Elimination mit Spaltenpivotsuche.

1. "Pivotsuche liefert immer genauere Ergebnisse"

   Die Matrix

   $$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$

   ist singulär, was eine Gauß–Elimination ohne *Pivotsuche auch erkennen würde, die (mit Überschreiben) die Folge*

   $$A^{(1)} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & -1 & -2 \\ 3 & -2 & -4 \end{bmatrix} \qquad \rightarrow \qquad A^{(2)} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & -1 & -2 \\ 3 & 2 & 0 \end{bmatrix}$$

   liefert. Duch die Pivotsuche wird aber zuerst durch 3 *geteilt und die dabei entstehenden Rundungsfehler sorgen dafür, daß die Matrix "ein bißchen invertierbar" aber lausig konditioniert wird.*

2. "Durch Pivotsuche wird das Produkt $|L|\,|U|$ kleiner"

   Wir betrachten die Matrix

   $$\begin{bmatrix} \frac{9}{10} & 0 & 0 \\ 1 & 100 & 100 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ \frac{10}{9} & 1 & \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{9}{10} & 0 & 0 \\ & 100 & 100 \\ & & 1 \end{bmatrix}.$$

Insbesondere ist $A = |L|\,|U|$. Mit Pivotsuche müssen wir die Zeilen 1 und 2 vertauschen und erhalten die LU–Zerlegung

$$\begin{bmatrix} 1 & 100 & 100 \\ \frac{9}{10} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ \frac{9}{10} & 1 & \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 100 & 100 \\ & -90 & -90 \\ & & 1 \end{bmatrix}$$

und

$$|L|\,|U| = \begin{bmatrix} 1 & & \\ \frac{9}{10} & 1 & \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 100 & 100 \\ & 90 & 90 \\ & & 1 \end{bmatrix} = \begin{bmatrix} 1 & 100 & 100 \\ \frac{9}{10} & 180 & 180 \\ 0 & 0 & 1 \end{bmatrix}.$$

Merke: Zeilenpivotsuche sorgt nur dafür, daß alle Einträge von $|L|$ einen Wert ≤ 1 haben. $|U|$ kann dafür ganz nett wachsen.

4.6 Verbesserung der Genauigkeit – Skalierung

Eine typische Fehlerquelle für lineare Gleichungssysteme besteht in schlechter *Skalierung*, bei der verschiedenen Gleichungen in verschiedenen Skalen (beispielsweise in m und cm) angegeben werden, was zu Gleichungssystemen der Form

$$\begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix} x = \begin{bmatrix} 100 \\ 1 \end{bmatrix}$$

führt – die Lösung ist natürlich $[1,1]^T$, aber die Gleichungen sind schlecht skaliert. Der Ansatz zur Behebung dieses Problems, die *Skalierung*, basiert auf der Beobachtung, daß für beliebige Diagonalmatrizen $D_1, D_2 \in \mathbb{R}_+^{n\times n}$

$$Ax = b \qquad \Longleftrightarrow \qquad \underbrace{D_1 A D_2^{-1}}_{=:B}\underbrace{D_2 x}_{=:y} = \underbrace{D_1 b}_{=:c} \qquad \Longleftrightarrow \qquad By = c. \tag{4.42}$$

Man löst also zuerst $By = c$ und bestimmt dann $\hat{x} = D_2^{-1}\hat{y}$; sind die Diagonaleinträge in D_1 und D_2 Potenzen der Basis B der Rechnerarithmetik, dann können die zusätzlichen Operationen *exakt* ausgeführt werden, es gibt also keine weiteren Rundungsfehler.

Die Bedeutung dieses Ansatzes liegt darin, daß man durch "cleveres" Skalieren die Konditionszahl einer Matrix deutlich verbessern kann.

Beispiel 4.38 *Wir betrachten, für $\alpha \in \mathbb{R}_+$, die (zeilenskalierte) Matrix*

$$A = \begin{bmatrix} \alpha & 1000\,\alpha \\ 1 & 1 \end{bmatrix} \qquad \Longrightarrow \qquad \|A\|_\infty = \max\{1001\,\alpha, 2\},$$

mit Inverser

$$A^{-1} = \begin{bmatrix} -\frac{1}{999\,\alpha} & \frac{1000}{999} \\ \frac{1}{999\,\alpha} & -\frac{1}{999} \end{bmatrix} \qquad \Longrightarrow \qquad \left\|A^{-1}\right\|_\infty = \frac{1000 + \alpha^{-1}}{999}.$$

Damit ist

$$\kappa_\infty(A) = \begin{cases} \dfrac{1001}{999}(1000\,\alpha + 1), & \alpha \geq 2/1001, \\ \dfrac{2000 + 2\alpha^{-1}}{999}, & \alpha < 2/1001, \end{cases} \tag{4.43}$$

was für $\alpha = 2/1001$ *den minimalen Wert*

$$\kappa_\infty(A) = \frac{3001}{999} \sim 3$$

annimmt. Für $\alpha = 1$ *ist die Konditionszahl hingegen* > 1000*!*
Allerdings ist es nicht einfach, die beste Skalierung zu finden: Der "naive" Ansatz hätte wohl $\alpha = 1/1000$*, also*

$$A = \begin{bmatrix} 1/1000 & 1 \\ 1 & 1 \end{bmatrix}$$

gewählt, was eine nicht optimale[29] *Konditionszahl von* ~ 4 *geliefert hätte. Nicht schlecht aber halt auch nicht optimal!*

Es gibt auch komplexere Pivotstrategien, die zu impliziter Skalierung führen. Man bestimmt (bei Zeilenvertauschungen) das m–te Pivotelement (also eine Zeile) durch

$$\frac{\left|a_{jm}^{(m)}\right|}{\max\limits_{r=m,\ldots,n}\left|a_{jr}^{(m)}\right|} = \max_{k=m,\ldots,n} \frac{\left|a_{km}^{(m)}\right|}{\max\limits_{r=m,\ldots,n}\left|a_{kr}^{(m)}\right|}. \tag{4.44}$$

Satz 4.39 *(J. M. Peña)*
Ist A *eine invertierbare Matrix mit* $PA = LU$ *und hat diese* LU*–Zerlegung die Eigenschaft, daß* $LU = |L|\,|U|$*, dann wird sie von Gauß–Elimination mit der Pivotstrategie (4.44) geliefert.*

4.7 Verbesserung der Genauigkeit – Iteratives Verfeinern

Die zweite Methode zur Verbesserung der Genauigkeit des Ergebnisses basiert auch wieder auf einer einfachen Idee: ausgehend von einer berechneten Lösung $\hat{x}$ sehen wir uns das *Residuum*

$$r_1 = b - A\hat{x}.$$

an. Ist der berechnete Werte $\hat{r}_1$ so gut, daß $|\hat{r}_1| \leq n\hat{u}\,|1_n|$[30], dann können wir nichts besseres erwarten, andernfalls berechnen wir eine *Korrektur* $\hat{y}_1$ als Lösung von $Ay = \hat{r}_1$, setzen $\hat{x}_1 = \hat{x} + \hat{y}_1$ und betrachten

$$r_2 = b - A\hat{x}_1,$$

[29] Aber gar nicht so schlechte ...
[30] Die Lösung $\hat{x}$ ist also bis auf unvermeidbare Rundungsfehler exakt!

```
%%  LoesAxb3.m (Numerik 1)
%%  ------------------------------------------------------
%%  Loesen eines Gleichungssystems
%%  (mit Pivot und k-facher Nachiteration)
%%  Eingabe:
%%    A    Matrix
%%    b    rechte Seite
%%    k    % Nachiterationen

function x = LoesAxb3( A,b,k )
  [ LU,p ] = GaussSP( A );

  y = VorElim2( LU,b,p );
  x = RueckSubs( LU,y );

  for j = 1:k
    r = b - A*x;
    disp( norm(r) );

    y = VorElim2( LU,r,p );
    d = RueckSubs( LU,y );

    x = x+d;
  end
% endfunction
```

Programm 4.11 `LoesAxb3.m`: Gauß–Elimination mit texttt*k*–facher Nachiteration.

was wir solange fortsetzen bis (hoffentlich) irgendein r_k klein genug wird.
Iteratives Verfeinern[31] (*iterative refinement*) berechnet also, ausgehend von $\hat{x}_0 = \hat{x}$, eine Folge

$$\hat{r}_j = b \ominus A \otimes \hat{x}_{j-1}, \qquad \hat{x}_j = \hat{x}_{j-1} \oplus \texttt{Solve}\,(Ax = \hat{r}_j)\,, \qquad j \in \mathbb{N}, \tag{4.45}$$

in der Hoffnung, daß $\hat{r}_j \to 0$ (oder zumindest klein genug wird).

Bemerkung 4.40 *Beim "klassischen"* iterative refinement *werden die Operationen aus (4.45) in* doppelter *Genauigkeit, also mit einem relativen Fehler von höchstens* $2\hat{u}^2$*, berechnet.*

Was bringt uns nun die Nachiteration? Ganz einfach: Anstatt die komplexe Aufgabe des Lösens eines linearen Gleichungssystems in hoher Genauigkeit durchzuführen, tun wir das

[31] Auch als *Nachiteration* bekannt.

```
%%  VorElim2.m (Numerik 1)
%%  ------------------------------------------------------
%%  Vorwaertselimination mit Permutation, ueberschreibt b,
%%    Diagonale = 1!!!
%%  Eingabe:
%%    L    untere Dreiecksmatrix
%%    b    rechte Seite
%%    p    Permutationsvektor

function x = VorElim2( L,b,p )
  n = length( b );

  bb = b;
  for j = 1:n
    bb( j ) = b( p(j) );
  end
  b = bb;

  for j = 1:n
    b(j) = b(j) - L(j,1:j-1) * b( 1:j-1 );
  end

  x = b;
endfunction
```

Programm 4.12 `VorElim2.m`: Vorwärtselimination mit “integrierter” Permutation.

schneller[32] und nehmen einen gewissen Fehler in Kauf, der dann mit der Nachiteration beseitigt wird. Und bei der Nachiteration müssen nur Matrix–Vektor–Produkte berechnet werden, was deutlich billiger ist, als ein Gleichungssystem zu lösen.

Eine typische Fehlerabschätzung ist beispielsweise das folgende Resultat (als leichte Abschwächung einer Aussage aus dem Buch von Higham [20]).

Satz 4.41 *Erfüllt die invertierbare Matrix $A \in \mathbb{R}^{n\times n}$ die Bedingung $2(C+1)n\kappa_\infty(A) \ll \hat{u}^{-1}$, dann bestimmt iterative Verfeinerung ein $\hat{x}$ mit einer relative Genauigkeit*

$$\frac{\|\hat{x}-x\|_\infty}{\|x\|_\infty} \approx \begin{cases} \hat{u}, & \tilde{u} \sim \hat{u}^2, \\ 2(n+1)\hat{u}\|A\|_\infty, & \tilde{u} \sim \hat{u}. \end{cases} \tag{4.46}$$

[32] Es gibt Prozessoren, die in Spielekonsolen verwendet werden (Stand der Technologie: 2006), bei denen einfach–genaue Rechnung um den Faktor 25 schneller ist als doppelt–genaue, und da lohnt sich das dann schon.

Es wird also wahrscheinlich richtig sein, das Gattungsmäßige gemeinsam zu behandeln [. . .] Was aber nicht so ist, muß man einzeln behandeln.

Aristoteles, Die Methodik der Naturforschung.

5 Spezielle lineare Gleichungssysteme

Bevor wir uns spezielle Systeme ansehen, erst eine kleine Variation von Satz (4.32).

Satz 5.1 *Eine Matrix $A \in \mathbb{R}^{n\times n}$ ist genau dann invertierbar, wenn es eine Diagonalmatrix $D \in \mathbb{R}^{n\times n}$, eine Permutationsmatrix P, sowie eine untere Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ und eine obere Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$ gibt, so daß $\ell_{jj} = u_{jj} = 1$, $d_{jj} \neq 0$, $j = 1, \ldots, n$, und*

$$PA = LDU. \tag{5.1}$$

Beweis: Nach Satz (4.32) gibt es P, L, U' so daß $PA = LU'$ und $u'_{jj} \neq 0$, $j = 1, \ldots, n$. Dann setzen wir

$$D := \text{diag } [u_{jj} : j = 1, \ldots, n] \qquad \text{und} \qquad U = D^{-1}U',$$

was (5.1) liefert. □

5.1 Cholesky–Zerlegung

Die Cholesky–Zerlegung beschäftigt sich mit symmetrischen, (strikt) positiv definiten Matrizen.

Definition 5.2 *Eine Matrix $A \in \mathbb{R}^{n\times n}$ heißt* symmetrisch*, wenn $A^T = A$ und (strikt) positiv definit, wenn*

$$x^T Ax > 0, \qquad x \in \mathbb{R}^n \setminus \{0\}.$$

Ist lediglich $x^T Ax \geq 0$, dann nennt man A positiv semidefinit*.*

Bemerkung 5.3 *Die Terminologie zu positiver Definitheit ist gefährlich mehrdeutig! Zwei Punkte sind hier besonders zu beachten:*

1. *Man kann entweder wie in Definition 5.2 zwischen "positiv definit" und "positiv semidefinit" unterscheiden*[33]*, oder aber die Begriffe "strikt positiv definit" und "positiv definit" verwenden*[34]*. Deswegen empfielt es sich immer,* genau *zu prüfen, wie die Begriffe verwendet werden.*

2. *Oftmals beinhaltet die Definition von "positiv definit" auch bereits, daß die Matrix symmetrisch ist. Der Grund ist, daß für $A \in \mathbb{C}^{n\times n}$ die Bedingung*

$$x^T Ax \geq 0, \qquad x \in \mathbb{C}^n$$

insbesondere bedeutet, daß $x^T Ax$ eine reelle *Zahl ist, und das ist wiederum nur dann der Fall, wenn A eine* hermitesche *Matrix ist, also $A^H = A$ ist. Und für reelles A heißt das natürlich Symmetrie!*

Satz 5.4 *Ist $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit, dann gibt es eine untere Dreiecksmatrix $G \in \mathbb{R}^{n\times n}$, so daß*

$$A = GG^T. \tag{5.2}$$

Beweis: Zuerst bemerken wir, daß A genau dann positiv definit ist, wenn alle Hauptminoren A_m, $m = 1, \ldots, n$ positiv definit sind. Gäbe es nämlich ein $1 \leq m \leq n$ und ein $y \in \mathbb{R}^m$ so daß

$$0 \geq y^T A_m y = \begin{bmatrix} y^T & 0 \end{bmatrix} A \begin{bmatrix} y \\ 0 \end{bmatrix},$$

dann wäre dies ein Widerspruch zur positiven Definitheit von A. Insbesondere sind alle Hauptminoren von A invertierbar und damit gibt es, nach Satz 5.1, Dreiecksmatrizen L, U mit Diagonale 1 und eine Diagonalmatrix D, so daß

$$LDU = A = A^T = U^T DL^T$$

und wegen der Eindeutigkeit der LDU–Zerlegung ist $U = L^T$. Also ist

$$A = LDL^T$$

und L, D haben vollen Rang n. Für $j = 1, \ldots, n$ ist außerdem

$$0 < \left(L^{-T} e_j\right)^T A \left(L^{-T} e_j\right) = e_j^T L^{-1} LDL^T L^{-T} e_j = e_j^T De_j = d_{jj}.$$

Damit ist

$$D = EE^T, \qquad E = \text{diag}\left[\sqrt{d_{jj}} : j = 1, \ldots, n\right],$$

und $G := LE$ liefert die gewünschte Zerlegung (5.2). □

[33] Was meistens in der deutschsprachigen Literatur geschieht.

[34] Wie in der englischsprachigen Literatur. Richtig verwirrend wird es dann, wenn Deutsche in Englisch schreiben und ihre gewohnte Begriffswelt beibehalten.

```
%%  Cholesky.m (Numerik I)
%%  ------------------------------------------------------
%%  Berechnet Cholesky-Zerlegung
%%  Eingabe:
%%    A    Matrix

function G = Cholesky( A )
  n = length(A);
  G = zeros( n );

  for j = 1:n
    g = A( j,j ) - G( j,1:j-1 ).^2;
    G( j,j ) = sqrt( g );
    for k = j+1:n
      g = A( k,j ) - G( j,1:j-1 ) * G( k,1:j-1 );
      G( k,j ) = g / G( j,j );
    end
  end
%endfunction
```

Programm 5.1 `Cholesky.m`: Das Cholesky–Verfahren nach (5.4) und (5.5).

Um die *Cholesky–Zerlegung* $A = GG^T$ zu bestimmen, gehen wir genau wie beim Doolittle–Verfahren vor und nutzen die Symmetrie von A und die Beziehung

$$a_{jk} = \sum_{r=1}^{j} g_{jr} g_{kr}, \qquad 1 \le j \le k \le n, \tag{5.3}$$

aus. Das liefert uns, daß

$$g_{jj} = \sqrt{a_{jj} - \sum_{r=1}^{j-1} g_{jr}^2}, \qquad j = 1, \ldots, n, \tag{5.4}$$

$$g_{kj} = \left(a_{kj} - \sum_{r=1}^{j-1} g_{jr} g_{kr} \right) / g_{jj}, \qquad 1 \le j < k \le n. \tag{5.5}$$

5.2 Bandierte Systeme

Eine Matrix $A \in \mathbb{R}^{n \times n}$ wird als (p, q)–*bandiert* bezeichnet, wenn

$$a_{jk} = 0, \qquad j > k + p, \quad k > j + q. \tag{5.6}$$

Der Speicheraufwand für eine (p,q)–bandierte Matrix ist nur $O\left((p+q+1)n\right)$[35], das heißt, für $p+q \ll n$ kann man solche Matrizen sehr ökonomisch speichern. Und diese Ökonomie würde man gerne auf die LU–Zerlegung übertragen.

Satz 5.5 *Besitzt die (p,q)–bandierte Matrix $A \in \mathbb{R}^{n\times n}$ eine LU–Zerlegung $A = LU$, dann ist L eine $(p,0)$–bandierte Matrix und U eine $(0,q)$–bandierte Matrix.*

Bemerkung 5.6 *(LU–Zerlegung für bandierte Matrizen)*

1. *Es ist (leider) klar, daß eine Zerlegung wie in Satz 5.5 nicht mehr existiert, wenn man Zeilen– und/oder Spaltenvertauschungen durchführen muß, da dies normalerweise die Bandstruktur der Matrix zerstören wird.*

2. *Besonders gut geeignet für so eine Zerlegung sind Matrizen, die offensichtlich eine LU–Zerlegung besitzen, z.B. strikt (Spalten–) diagonaldominante Matrizen.*

Beweis von Satz 5.5: Induktion über n; $n = 1$ ist klar. Sei also $A \in \mathbb{R}^{n+1\times n+1}$ eine (p,q)–Bandmatrix, geschrieben als

$$A = \begin{bmatrix} a & w^T \\ v & B \end{bmatrix}, \qquad a \neq 0,\ v,w \in \mathbb{R}^n,\ B \in \mathbb{R}^{n\times n},$$

wobei $v_{p+1} = \cdots = v_n = 0$, $w_{q+1} = \cdots = w_n = 0$, und B eine (p,q)–bandierte Matrix ist. Nach einem Schritt Gauß–Elimination ist dann

$$A = \begin{bmatrix} 1 & 0 \\ \frac{v}{a} & I_n \end{bmatrix} \begin{bmatrix} a & w^T \\ 0 & B - \frac{vw^T}{a} \end{bmatrix}.$$

Da

$$vw^T = \begin{bmatrix} v_1w_1 & \cdots & v_1w_q & \\ \vdots & \ddots & \vdots & \\ v_pw_1 & \cdots & v_pw_q & \\ & & & 0 \end{bmatrix}$$

ebenfalls (p,q)–bandiert ist, besitzt die (p,q)–bandierte Matrix $B - vw^T/a$ eine $(p,0)$- bzw. $(0,q)$–bandierte LU–Zerlegung

$$B - \frac{vw^T}{a} = \tilde{L}\tilde{U},$$

und daher ist

$$A = \underbrace{\begin{bmatrix} 1 & 0 \\ v & \tilde{L} \end{bmatrix}}_{=:L} \underbrace{\begin{bmatrix} a & w^T \\ 0 & \tilde{U} \end{bmatrix}}_{=:U},$$

wobei L eine $(p,0)$–bandierte Matrix und U eine $(0,q)$–bandierte Matrix ist. □

Eine *bandierte* Version der Gauß–Elimination findent sich in `BGauss.m`.

[35] Für Erbsenzähler: Es sind $n^2 - \frac{(n-p)(n-p-1)}{2} - \frac{(n-q)(n-q-1)}{2}$ Einträge

```
%%  BGauss.m (Numerik I)
%%  -----------------------------------------------------
%%  Gauss-Elimination fuer bandierte Matrizen
%%  (ohne Pivot, aber mit Matlab-Features)
%%  Eingabe:
%%    A    Matrix
%%    p,q  Bandbreite

function LU = BGauss( A,p,q )
  n = length( A );

  for m = 1:n
    for j = m+1:min( n,m+p )
      A( j,m ) = A( j,m ) / A( m,m );
      r = min( n,m+q );
      A( j,m+1:r ) = A( j,m+1:r ) - A( j,m ) * A( m,m+1:r );
    end
  end
  LU = A;
%endfunction
```

Programm 5.2 `BGauss.m`: Gauß–Elimination für bandierte Systeme mit Aufwand $O\left(n^2\right)$ solange $p, q \ll n$.

5.3 Total positive Systeme

Total positive Matrizen verhalten sich, was die Gauß–Elimination angeht, besonders brav und tauchen noch dazu in wichtigen Anwendungen auf. Das Standardwerk zu diesem Thema ist mit Sicherheit das Buch von Karlin [23].

Definition 5.7 *Für $A \in \mathbb{R}^{n\times n}$ und $I, J \subseteq \{1,\dots,n\}$ mit $\#I = \#J\ (= k\ )$ bezeichnet*

$$A(I,J) = A\begin{pmatrix} i_1,\dots,i_k \\ j_1,\dots,j_k \end{pmatrix} = \begin{bmatrix} a_{i_1,j_1} & \dots & a_{i_1,j_k} \\ \vdots & \ddots & \vdots \\ a_{i_k,j_1} & \dots & a_{i_k,j_k} \end{bmatrix}, \qquad \begin{matrix} i_1 < \dots < i_k, \\ j_1 < \dots < j_k, \end{matrix}$$

die durch I, J indizierte quadratische Teilmatrix, bei der Zeilen und Spalten aber immer noch in derselben Reihenfolge *angeordnet sind wie in A.*
Außerdem verwenden wir die Abkürzung $N = \{1,\dots,n\}$, also

$$A = A(N,N).$$

Definition 5.8 *Eine Matrix $A \in \mathbb{R}^{n\times n}$ heißt* total positiv*, wenn*

$$\det A(I,J) > 0, \qquad I,J \subseteq \{1,\dots,n\}, \quad \#I = \#J, \tag{5.7}$$

und total nichtnegativ*, wenn*

$$\det A(I,J) \geq 0, \qquad I,J \subseteq \{1,\dots,n\}, \quad \#I = \#J, \tag{5.8}$$

Bemerkung 5.9 *Ist eine Matrix A total nichtnegativ, dann ist*

$$a_{jk} \geq 0, \qquad j,k = 1,\dots,n.$$

Beispiel 5.10 *Die* Pascal–Matrizen

$$P_n := \left[\binom{j+k-2}{j-1} : j,k = 1,\dots,n\right]$$

sind symmetrisch und total positiv. Sie sind außerdem exponentiell schlecht konditioniert:

$$\kappa_2(P_n) \sim \frac{16^n}{n\pi}.$$

Die Bedeutung total nichtnegativer Matrizen für Eliminationsverfahren ergibt sich aus dem nächsten Satz.

Satz 5.11 *Ist $A \in \mathbb{R}^{n\times n}$ eine invertierbare und total nichtnegative Matrix, dann gibt es eine untere Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ und eine obere Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$, so daß*

$$A = LU \qquad \textit{und} \qquad L \geq 0, \quad U \geq 0. \tag{5.9}$$

Bemerkung 5.12 *Kombiniert man Satz 5.5 mit Satz 5.11, so sind total nichtnegative, stark bandierte (d.h., $p+q \ll n$) Matrizen besonders effektiv und stabil mit Gauß–Elimination lösbar. Und solche Matrizen treten in der Praxis tatsächlich auf, nämlich bei der* Splineinterpolation*, siehe Satz 8.6.*

Lemma 5.13 *Ist $A \in \mathbb{R}^{n\times n}$ total nichtnegativ und invertierbar, dann ist $a_{11} > 0$.*

Beweis: Angenommen, es wäre $a_{11} = 0$ und $a_{1k} > 0$ für ein $k > 1$. Da A total nichtnegativ ist, ist für jedes $m > 1$

$$0 \le \det A\left(\{1,m\},\{1,k\}\right) = \begin{vmatrix} a_{11} & a_{1k} \\ a_{m1} & a_{mk} \end{vmatrix} = -a_{1k}a_{m1} \qquad \Rightarrow \qquad a_{m1} = 0.$$

Das heißt, wenn $a_{11} = 0$ ist, dann ist entweder $a_{1k} = 0$, $k = 1,\ldots,n$, oder $a_{k1} = 0$, $k = 1,\ldots,n$, oder sogar beides, aber auf alle Fälle ist dann A nicht invertierbar. □

Lemma 5.14 *Sei $A \in \mathbb{R}^{n\times n}$ total nichtnegativ und invertierbar. Dann ist die Matrix $\widetilde{A}$, die man nach einem Gauß–Eliminationsschritt als*

$$A \to A^{(1)} := \begin{bmatrix} a_{11} & v^T \\ 0 & \widetilde{A} \end{bmatrix}, \qquad \widetilde{A} = [\tilde{a}_{jk} : j,k = 2,\ldots,n] \in \mathbb{R}^{n-1\times n-1}.$$

erhält, auch total nichtnegativ und invertierbar.

Beweis: Für beliebige $I, J \subseteq \{2,\ldots,n\}$ betrachten wir

$$\det \widetilde{A}\,(I,J) = \frac{1}{a_{11}} \det A^{(1)}\left(I\cup\{1\}, J\cup\{1\}\right). \tag{5.10}$$

Mit

$$A = \begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix} \qquad \text{und} \qquad \alpha_j = \frac{a_{j1}}{a_{11}}, \quad j = 2,\ldots,n,$$

ist

$$A^{(1)} = \begin{bmatrix} a_1^T \\ a_2^T - \alpha_2 a_1^T \\ \vdots \\ a_n^T - \alpha_n a_1^T \end{bmatrix}$$

und damit

$$A^{(1)}\left(I\cup\{1\}, J\cup\{1\}\right) = \begin{bmatrix} a_1^T\left(J\cup\{1\}\right) \\ a_{i_1}^T\left(J\cup\{1\}\right) - \alpha_{i_1} a_1^T\left(J\cup\{1\}\right) \\ \vdots \\ a_{i_k}^T\left(J\cup\{1\}\right) - \alpha_{i_k} a_1^T\left(J\cup\{1\}\right) \end{bmatrix}. \tag{5.11}$$

Da die Determinante invariant unter Addition von Vielfachen von Zeilen ist, haben wir also, daß

$$\det A^{(1)}\left(I \cup \{1\}, J \cup \{1\}\right) = \begin{vmatrix} a_1^T\left(J \cup \{1\}\right) \\ a_{i_1}^T\left(J \cup \{1\}\right) \\ \vdots \\ a_{i_k}^T\left(J \cup \{1\}\right) \end{vmatrix} = \det A\left(I \cup \{1\}, J \cup \{1\}\right),$$

und setzen wir dies in (5.10) ein, dann erhalten wir, daß

$$\det \widetilde{A}\left(I, J\right) = \frac{1}{a_{11}} \det A\left(I \cup \{1\}, J \cup \{1\}\right), \qquad I, J \subset \{2, \ldots, n\}, \quad \#I = \#J. \tag{5.12}$$

Damit ist $\widetilde{A}$ total nichtnegativ und außerdem ist

$$\det \widetilde{A} = \frac{1}{a_{11}} \det A > 0$$

und damit ist $\widetilde{A}$ auch invertierbar. □

Beweis von Satz 5.11: Induktion über n, der Fall $n = 1$ ist trivial. Für beliebiges $n > 1$ machen wir wieder einen Schritt Gauß–Elimination, erhalten

$$A^{(1)} = \begin{bmatrix} a_{11} & A\left(\{1\}, \{2, \ldots, n\}\right) \\ 0 & \widetilde{A} \end{bmatrix},$$

wenden die Induktionshypothese auf $\widetilde{A}$ an (das geht nach Lemma 5.14) und erhalten $\widetilde{L} \geq 0$ und $\widetilde{U} \geq 0$ so daß $\widetilde{A} = \widetilde{L}\widetilde{U}$. Dann ist aber

$$A = \underbrace{\begin{bmatrix} 1 & 0 \\ a_{11}^{-1} A\left(\{2, \ldots, n\}, 1\right) & \widetilde{L} \end{bmatrix}}_{=:L} \underbrace{\begin{bmatrix} a_{11} & A\left(\{1\}, \{2, \ldots, n\}\right) \\ 0 & \widetilde{U} \end{bmatrix}}_{=:U},$$

und es ist $L, U \geq 0$. □

Korollar 5.15 *Die LU–Zerlegung aus (5.9) erhält man durch Gauß–Elimination* ohne *Pivotsuche (oder durch Doolittle).*

Da nun

$$\left|\widehat{L}\right| \left|\widehat{U}\right| \leq |L| \ |U| + \left(\left|\widehat{L} - L\right| + \left|\widehat{U} - U\right|\right) = \left(1 + O\left(\hat{u}^2\right)\right) |A|,$$

ist auch

$$\left\| \left|\widehat{L}\right| \left|\widehat{U}\right| \right\|_\infty = \left(1 + O\left(\hat{u}^2\right)\right) \|A\|_\infty$$

und wie erhalten die folgende, sehr gute Fehleraussage.

Korollar 5.16 *Es sei $A \in \mathbb{R}^{n\times n}$ total nichtnegativ, invertierbar und erfülle*

$$\kappa_\infty(A) \ll \hat{u}^{-1}.$$

Dann gilt für die mit Gauß–Elimination ohne Pivotsuche berechnete Lösung $\hat{x}$ zu $Ax = b$ die Abschätzung

$$\frac{\|\hat{x} - x\|_\infty}{\|x\|_\infty} \leq \frac{8n\hat{u}}{1 - \hat{u}\kappa_\infty(A)}\, \kappa_\infty(A). \tag{5.13}$$

Anstelle von Satz 5.11 gilt sogar die folgende, schärfere Aussage.

Satz 5.17 *Eine Matrix $A \in \mathbb{R}^{n\times n}$ ist genau dann invertierbar und total nichtnegativ, wenn es* total nichtnegative *Dreiecksmatrizen $L, U \in \mathbb{R}^{n\times n}$, $\ell_{jj} = 1$, $u_{jj} \neq 0$, $j = 1, \ldots, n$, gibt, so daß $A = LU$.*

Nur eine Weile muß vergehn;
Dann ist auch dieses überstanden.
Dann wird mit hell euch zugewandten
Augen das Neue vor euch stehn.

Joachim Ringelnatz, "Über eine Weile–"

6 Iterative Verfahren für lineare Gleichungssysteme

Wir betrachten wieder eine invertierbare Matrix $A \in \mathbb{R}^{n\times n}$ und ein Gleichungssystem

$$Ax = b.$$

Ein *iteratives Verfahren* besteht aus einer Berechnungsvorschrift

$$x^{(j+1)} = F\left(x^{(j)}\right)$$

mit der Hoffnung, daß $x^{(j)} \to x$ für $j \to \infty$. Im Gegensatz zu den direkten Verfahren berechnen wir also keine "exakte" Lösung[36] mehr, sondern versuchen, uns näher und näher an die Lösung "heranzutasten".

Ist x die Lösung von $Ax = b$, so gilt für beliebiges invertierbares $B \in \mathbb{R}^{n\times n}$

$$b = Ax = Bx + (A - B)\,x \qquad \Longrightarrow \qquad x = B^{-1}b + \left(I - B^{-1}A\right)x,$$

also ist x ein *Fixpunkt* von

$$F(x) = B^{-1}b + \left(I - B^{-1}A\right)x. \tag{6.1}$$

Damit haben wir das Problem "Lösen eines linearen Gleichungssystems" also in ein Problem der Form "Bestimme den Fixpunkt einer Funktion" umgewandelt. Das hat dann auch einen ganz unmittelbaren Vorteil: Wenn ein Iterationswert $x^{(j)}$ eine *exakte Lösung* des Problems, also ein Fixpunkt ist, dann wird die Funktion F diesen in sich selbst abbilden – das Lösungsverfahren "verwirft" einmal gefundene Lösungen nicht wieder.

Die Fixpunktsuche kann aber eine recht einfache und automatische Sache sein, wenn F eine bestimmte Eigenschaft hat.

Definition 6.1 *Eine Abbildung $F : \mathbb{R}^n \to \mathbb{R}^n$ heißt* Kontraktion *bezüglich einer Norm $\|\cdot\|$, wenn es eine Konstante $\rho < 1$ gibt, so daß*

$$\|F(x) - F(y)\| \le \rho\,\|x - y\|, \qquad x, y \in \mathbb{R}^n. \tag{6.2}$$

[36] Bei exakter Rechnung (z.B. in Computeralgebrasystemen) würden die direkten Verfahren ja nach endlich vielen Schritten eine exakte Lösung liefern.

Satz 6.2 *(Banachscher Fixpunktsatz für den $\mathbb{R}^n$)*
Ist $F : \mathbb{R}^n \to \mathbb{R}^n$ eine Kontraktion bezüglich einer Norm $\|\cdot\|$, dann hat F genau einen Fixpunkt x^* und für jedes $x^{(0)}$ konvergiert die Folge

$$x^{(j+1)} = F\left(x^{(j)}\right), \qquad j \in \mathbb{N}_0, \tag{6.3}$$

gegen x^*.

Beweis: Beginnen wir mit der Eindeutigkeit: wären x^*, y^* zwei verschiedene Fixpunkte, also $\|x^* - y^*\| > 0$, dann wäre

$$1 = \frac{\|x^* - y^*\|}{\|x^* - y^*\|} = \frac{\|F(x^*) - F(y^*)\|}{\|x^* - y^*\|} \leq \frac{\rho \, \|x^* - y^*\|}{\|x^* - y^*\|} = \rho,$$

ein Widerspruch zu $\rho < 1$.
Jetzt zur Existenz. Für $j \geq 1$ ist

$$\left\|x^{(j+1)} - x^{(j)}\right\| = \left\|F\left(x^{(j)}\right) - F\left(x^{(j-1)}\right)\right\| \leq \rho \left\|x^{(j)} - x^{(j-1)}\right\| \leq \cdots \leq \rho^j \left\|x^{(1)} - x^{(0)}\right\|$$

und daher, für $N > 0$,

$$\begin{aligned} \left\|x^{(j+N)} - x^{(j)}\right\| &\leq \sum_{k=1}^{N} \left\|x^{(j+k)} - x^{(j+k-1)}\right\| \leq \sum_{k=0}^{N-1} \rho^{j+k} \left\|x^{(1)} - x^{(0)}\right\| \\ &\leq \left\|x^{(1)} - x^{(0)}\right\| \rho^j \sum_{k=0}^{\infty} \rho^k = \frac{\rho^j}{1-\rho} \left\|x^{(1)} - x^{(0)}\right\|. \end{aligned}$$

Also ist $x^{(j)}$, $j \in \mathbb{N}_0$, eine Cauchyfolge und konvergiert somit gegen einen Grenzwert x^*, für den für beliebiges $j \in \mathbb{N}_0$ die Ungleichung

$$\|F(x^*) - x^*\| \leq \left\|F(x^*) - F\left(x^{(j)}\right)\right\| + \left\|x^* - x^{(j+1)}\right\| \leq \rho \left\|x^* - x^{(j)}\right\| + \left\|x^* - x^{(j+1)}\right\|,$$

gilt und da die rechte Seite für $j \to \infty$ gegen 0 konvergiert, ist $F(x^*) = x^*$. □

Jetzt aber zurück zu (6.1). Das daraus entstehende Lösungsverfahren wird also die Iteration

$$x^{(j+1)} = B^{-1}b + \left(I - B^{-1}A\right) x^{(j)} \tag{6.4}$$

verwenden. Die dabei verwendete Funktion F ist offensichtlich genau dann eine Kontraktion, wenn die *Iterationsmatrix* $(I - B^{-1}A)$ eine Kontraktion liefert. Die Ziele dieses Kapitels sind also:

- Bestimmung von "guten" Matrizen B (einfach zu invertieren und einfach aus A zu gewinnen).
- Beschreibung, wann $(I - B^{-1}A)$ eine Kontraktion ist.
- Bestimmung von Matrizen A (und passendem B), so daß (6.4) konvergiert.

Bemerkung 6.3 *Wenn die Iteration (6.4) gegen x^* konvergiert, dann ist*

$$x^* = B^{-1}b + \left(I - B^{-1}A\right) x^*.$$

Dies gilt sogar für jede *lineare Iteration der Form*

$$x^{(j+1)} = F\left(x^{(j)}\right) = Gx^{(j)} + g$$

mit $G \in \mathbb{R}^{n\times n}$ und $g \in \mathbb{R}^n$. Schließlich ist ja für jedes $j \in \mathbb{N}$

$$\begin{aligned} \left\|x^* - F\left(x^*\right)\right\| &\leq \left\|x^* - x^{(j+1)}\right\| + \left\|F\left(x^*\right) - x^{(j+1)}\right\| \\ &= \left\|x^* - x^{(j+1)}\right\| + \left\|F\left(x^* - x^{(j)}\right)\right\| \leq \left\|x^* - x^{(j+1)}\right\| + \|G\| \left\|x^* - x^{(j)}\right\| \end{aligned}$$

und da die rechte Seite für $j \to \infty$ gegen Null konvergiert während die linke Seite von j unabhängig ist, muß $x^* = F\left(x^*\right)$ sein.

6.1 Der Spektralradius und ein allgemeines Konvergenzkriterium

Der entscheidende Begriff für iterative Verfahren ist der des *Spektralradius*.

Definition 6.4 *Das* Spektrum $S(A)$ *einer Matrix $A \in \mathbb{R}^{n\times n}$ ist die Menge*

$$S(A) = \{\lambda \in \mathbb{C} \ : \ \ker{}_{\mathbb{C}^n}\left(A - \lambda I\right) \neq \{0\}\}$$

ihrer Eigenwerte. Der Spektralradius $\rho(A)$ *ist definiert als*

$$\rho(A) = \max_{\lambda\in S(A)} |\lambda| .$$

Proposition 6.5 *Es bezeichne $\|\cdot\|$ eine beliebige Vektornorm auf dem $\mathbb{R}^n$ sowie die zugehörige Operatornorm. Der Spektralradius $\rho(A)$ bestimmt sich als*

$$\rho(A) = \limsup_{j\to\infty} \left\|A^j\right\|^{1/j} \tag{6.5}$$

und erfüllt somit $\rho(A) \leq \|A\|$.

Beweis: Wir erweitern das Problem auf $\mathbb{C}^n$, es sei also jetzt $\|\cdot\|$ eine Fortsetzung der Vektornorm auf $\mathbb{C}^n$[37].
Sei $\lambda \in S(A)$ und $x \in \mathbb{C}^n$ ein dazugehöriger Eigenvektor mit $\|x\| = 1$. Dann ist

$$\left\|A^j\right\| \geq \left\|A^j x\right\| = |\lambda|^j \ \|x\| = |\lambda|^j$$

und damit ist

$$\limsup_{j\to\infty} \left\|A^j\right\|^{1/j} \geq \rho(A).$$

[37]Die p–Normen lassen sich beispielsweise ohne Änderung auf $\mathbb{C}^n$ übertragen, nur spielt der Betrag jetzt eine etwas wesentlichere Rolle.

Für die Umkehrung gibt es eine invertierbare Matrix $S \in \mathbb{C}^{n\times n}$, die A in ihre *Jordan–Normalform*

$$S^{-1}AS =: J = \begin{bmatrix} \lambda_1 & * & & \\ & \ddots & \ddots & \\ & & \lambda_{n-1} & * \\ & & & \lambda_n \end{bmatrix}, \qquad |\lambda_1| \le \cdots \le |\lambda_n|, \qquad * \in \{0,1\}.$$

überführt. Für jedes $\varepsilon > 0$ sei außerdem

$$D_\varepsilon = \begin{bmatrix} 1 & & & \\ & \varepsilon & & \\ & & \ddots & \\ & & & \varepsilon^{n-1} \end{bmatrix},$$

dann ist

$$D_\varepsilon J D_\varepsilon^{-1} = \begin{bmatrix} \lambda_1 & * & & \\ & \ddots & \ddots & \\ & & \lambda_{n-1} & * \\ & & & \lambda_n \end{bmatrix}, \qquad * \in \{0,\varepsilon\}$$

und daher

$$\left\| D_\varepsilon J D_\varepsilon^{-1} \right\| = |\lambda_n| + f(\varepsilon), \qquad f \in C(\mathbb{R}),\ f(0) = 0.$$

Somit ist, für festes $\varepsilon > 0$,

$$\begin{aligned} \left\|A^j\right\| &\le \|S\| \left\| \left(S^{-1}AS\right)^j \right\| \left\|S^{-1}\right\| = \kappa(S) \left\|J^j\right\| \le \kappa(S)\, \kappa\left(D_\varepsilon\right) \left\| \left(D_\varepsilon J D_\varepsilon^{-1}\right)^j \right\| \\ &\le \kappa(S)\, \kappa\left(D_\varepsilon\right) \left\| D_\varepsilon J D_\varepsilon^{-1} \right\|^j = \kappa(S)\, \kappa\left(D_\varepsilon\right) \left(|\lambda_n| + f(\varepsilon)\right)^j, \end{aligned}$$

also ist

$$\limsup_{j\to\infty} \left\|A^j\right\|^{1/j} \le \limsup_{j\to\infty} \underbrace{\left(\kappa(S)\, \kappa\left(D_\varepsilon\right)\right)^{1/j}}_{=1} \left(|\lambda_n| + f(\varepsilon)\right),$$

und damit

$$\limsup_{j\to\infty} \left\|A^j\right\|^{1/j} \le \max_{\lambda\in S(A)} |\lambda| + f(\varepsilon),$$

was mit $\varepsilon \to 0$ den Beweis vervollständigt. □

Bemerkung 6.6 *(Spektralradius)*

1. *Es ist schon wichtig, komplexe Eigenwerte mitzuberücksichtigen: die Matrix*

$$A = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

hat die (konjugiert) komplexen Eigenwerte $\pm i$, ihr reelles *Spektrum ist hingegen leer. Trotzdem ist $\rho(A) = 1$, was man auch aus (6.5) erhält, da $A^2 = -I$ ist.*

2. *Der Vorteil von (6.5) besteht darin, daß wir hiermit den Spektralradius einer Matrix mit "reellen Mitteln" bestimmen können.*

3. *Außerdem ist (6.5) von der gewählten Norm unabhängig. Dies kann man auch ganz einfach direkt einsehen, indem man sich daran erinnert, daß auf dem $\mathbb{R}^n$ alle Normen äquivalent sind.*

Nun aber zum angekündigten Konvergenzkriterium.

Satz 6.7 *Das Iterationsverfahren (6.4) konvergiert genau dann für jeden Startvektor $x^{(0)} \in \mathbb{C}^n$, wenn*

$$\rho\left(I - B^{-1}A\right) < 1. \tag{6.6}$$

Beweis: Ist (6.6) erfüllt, dann gibt es einen Index $m > 0$, so daß $\left\|(I - B^{-1}A)^m\right\| < 1$, das heißt, die Abbildung

$$F(x) := \left(I - B^{-1}A\right)^m x$$

ist eine Kontraktion und konvergiert, nach Satz 6.2 gegen einen *eindeutigen* Fixpunkt x^*, d.h., für $j = 0, \ldots, m-1$ gilt

$$F^k\left(x^{(j)}\right) = x^{(j+km)} \to x^*,$$

also konvergiert auch die ganze Folge.
Nehmen wir umgekehrt an, daß das Iterationsverfahren (6.4) gegen $x^* \in \mathbb{C}^n$ konvergiert und sei $y \in \mathbb{C}^n$ ein Eigenvektor von $(I - B^{-1}A)$ zum betragsgrößten Eigenwert λ. Setzen wir $x^{(0)} = x^* + y$, dann ist

$$x^{(1)} - x^* = \left(I - B^{-1}A\right)x^{(0)} + B^{-1}b - \left(I - B^{-1}A\right)x^* - B^{-1}b = \left(I - B^{-1}A\right)y = \lambda y,$$

sowie allgemein

$$x^{(j)} - x^* = \lambda^j y,$$

und Konvergenz liefert $|\lambda| < 1$. □

Für unsere Zwecke ist die folgende reelle Aussage hin– und ausreichend.

Korollar 6.8 *Das Iterationsverfahren (6.4) konvergiert für jeden Startvektor $x^{(0)} \in \mathbb{R}^n$ wenn*

1. $\rho\left(I - B^{-1}A\right) < 1$.
2. $\|I - B^{-1}A\| < 1$ *für eine Operatornorm* $\|\cdot\|$.

Man kann sogar Aussagen über die Konvergenzgeschwindigkeit machen und es dürfte nicht überraschend sein, daß die Konvergenz umso schneller ist, je kleiner der Spektralradius ist.

Satz 6.9 *Die Matrix $A \in \mathbb{R}^{n\times n}$ erfülle $\rho\left(I - B^{-1}A\right) < 1$ und es sei x die Lösung von $Ax = b$. Dann gilt, für jeden Startvektor $x^{(0)}$,*

$$\limsup_{j\to\infty}\left(\frac{\left\|x^{(j)} - x\right\|}{\|x^{(0)} - x\|}\right)^{1/j} \le \rho\left(I - B^{-1}A\right). \tag{6.7}$$

Beweis: Nach Proposition 6.5 gibt es für alle $\varepsilon \in (0, 1-\rho\left(I-B^{-1}A\right))$ einen Index $j_0 \in \mathbb{N}$ und eine Konstante $C > 0$, so daß

$$\left\|\left(I-B^{-1}A\right)^j\right\| \leq C\,\left(\varepsilon+\rho\left(I-B^{-1}A\right)\right)^j < 1, \qquad j > j_0,$$

und somit ist

$$\begin{aligned}\left\|x^{(j)}-x\right\| &= \left\|\left(I-B^{-1}A\right)^j\left(x^{(0)}-x\right)\right\| \leq \left\|\left(I-B^{-1}A\right)^j\right\|\,\left\|x^{(0)}-x\right\| \\ &\leq C\left(\varepsilon+\rho\left(I-B^{-1}A\right)\right)^j\left\|x^{(0)}-x\right\|, \qquad j > j_0,\end{aligned}$$

also

$$\limsup_{j\to\infty}\left(\frac{\left\|x^{(j)}-x\right\|}{\left\|x^{(0)}-x\right\|}\right)^{1/j} \leq \varepsilon+\rho\left(I-B^{-1}A\right)$$

und da ε beliebig war, folgt (6.7) □

6.2 Gauß–Seidel–Iteration und Jacobi–Iteration

Erinnern wir uns an unsere Anforderungen an die Matrix B für die *implizite* Iteration

$$Bx^{(j+1)} = b-(A-B)x^{(j)}.$$

Der einfachste Fall ist sicherlich

$$B = \text{diag}\,[a_{jj}\ :\ j=1,\ldots,n],$$

vorausgesetzt natürlich, daß $a_{jj} \neq 0$, $j = 1,\ldots,n$. Und das ist auch bereits das *Jacobi–Verfahren*, in der deutschsprachigen Literatur auch gerne als *Gesamtschrittverfahren* bezeichnet:

$$\begin{bmatrix} a_{11} & & \\ & \ddots & \\ & & a_{nn}\end{bmatrix} x^{(j+1)} = b - \begin{bmatrix} 0 & a_{12} & \ldots & a_{1n} \\ a_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n} \\ a_{n1} & \ldots & a_{n,n-1} & 0\end{bmatrix} x^{(j)} \tag{6.8}$$

beziehungsweise

$$x_k^{(j+1)} = \left(b_k - \sum_{r\neq k} a_{kr}x_r^{(j)}\right)/a_{kk}, \qquad k = 1,\ldots,n. \tag{6.9}$$

Bei der *Gauß–Seidel–Iteration*, dem *Einzelschrittverfahren*, setzt man

$$B = \begin{bmatrix} a_{11} & & 0 \\ \vdots & \ddots & \\ a_{n1} & \ldots & a_{nn}\end{bmatrix},$$

also

$$\begin{bmatrix} a_{11} & & 0 \\ \vdots & \ddots & \\ a_{n1} & \dots & a_{nn} \end{bmatrix} x^{(j+1)} = b - \begin{bmatrix} 0 & a_{12} & \dots & a_{1n} \\ & \ddots & \ddots & \vdots \\ & & \ddots & a_{n-1,n} \\ & & & 0 \end{bmatrix} x^{(j)}, \tag{6.10}$$

beziehungsweise

$$x_k^{(j+1)} = \left(b_k - \sum_{r=1}^{k-1} a_{kr} x_r^{(j+1)} - \sum_{r=k+1}^{n} a_{kr} x_r^{(j)} \right) / a_{kk}, \qquad k = 1, \dots, n. \tag{6.11}$$

Es gibt noch eine andere Interpretation von (6.11), nämlich daß bei der Bestimmung von $x_k^{(j+1)}$ *noch* die "alten" Komponenten $x_{k+1}^{(j)}, \dots, x_n^{(j)}$, aber *schon* die "neuen" Komponenten $x_1^{(j+1)}, \dots, x_{k-1}^{(j+1)}$ verwendet werden, die man ja gerade eben berechnet hat. Man verwendet also "alte" Werte nur noch dort, wo noch keine aktualisierten Werte vorhanden sind.
Was ist nun der Vorteil von solchen iterativen Verfahren?

1. Der Rechenaufwand je Iterationsschritt hängt *linear* von der Anzahl der Elemente $a_{jk} \neq 0$ ab. Damit sind solche Verfahren gut geeignet für große, dünnbesetzte Matrizen.

2. Wenn die Matrix "gutartig" ist (also das Iterationsverfahren schnell konvergiert), kann der Rechenaufwand weit besser als $O(n^3)$ sein: jeder Iterationsschritt hat ja "nur" $O(n^2)$ Operationen durchzuführen – und der Fehler fällt dabei immerhin wie $\rho(I - B^{-1}A)^j$, was bei gut konditionierten Matrizen[38] sehr schnell zu sehr guten Näherungen führen kann.

3. Das Jacobi–Verfahren ist noch dazu sehr gut parallelisierbar: Die Berechnungen der einzelnen Einträge von $x^{(j+1)}$ sind (nahezu) unabhängig voneinander, nur der Vektor $x^{(j)}$ muß "gespiegelt" werden. Für das Gauß–Seidel–Verfahren hingegen ist das Parallelisierungspotential sehr eingeschränkt.

Was aber helfen uns diese schönen Verfahren, wenn wir nicht wissen, ob und wann sie funktionieren. Natürlich kann man immer die Spektralradien testen, aber das ist nicht so einfach.

Satz 6.10 *Ist $A \in \mathbb{R}^{n \times n}$ symmetrisch und positiv definit, dann konvergiert das Gauß–Seidel–Verfahren.*

Beweis: Wir schreiben

$$A = \underbrace{L + D}_{=B} + \underbrace{L^T}_{=A-B},$$

wobei L eine *strikte* untere Dreiecksmatrix ist, das heißt, $\ell_{jj} = 0$, $j = 1, \dots, n$. Wir müssen nun also zeigen, daß

$$\rho(G) := \rho\left(B^{-1}(B - A)\right) = \rho\left(-(L + D)^{-1} L^T\right) < 1. \tag{6.12}$$

[38] Also solchen mit kleinem Spektralradius

```
%%  JacobiIt.m (Numerik 1)
%%  ------------------------------------------------------
%%  Jacobi-Iterationsschritt
%%  Eingabe:
%%    A    Matrix
%%    x    Naeherung fuer Loesung
%%    b    rechte Seite

function y = JacobiIt( A,x,b )
  n = length( A );
  y = zeros( n,1 );

  for j = 1:n
    z = b ( j ) - A( j,1:j-1 ) * x( 1:j-1 ) - A( j,j+1:n ) * x( j+1:n );
    y( j ) = z / A( j,j );
  end
%endfunction
```

Programm 6.1 `JacobiIt.m`: Eine Jacobi–Iteration nach (6.9).

Wegen der positiven Definitheit von A ist $d_{jj} = a_{jj} > 0$, $j = 1, \ldots, n$[39], und damit ist die Matrix G_1, definiert durch

$$\begin{aligned} G_1 &:= D^{1/2} G D^{-1/2} = -D^{1/2} (L+D)^{-1} L^T D^{-1/2} \\ &= -D^{1/2} (L+D)^{-1} D^{1/2} D^{-1/2} L^T D^{-1/2} \\ &= -\left(D^{-1/2} (L+D) D^{-1/2}\right)^{-1} (\underbrace{D^{-1/2} L D^{-1/2}}_{=:L_1})^T = -(I+L_1)^{-1} L_1^T \end{aligned}$$

wohldefiniert und *ähnlich* zu G und hat deswegen auch dieselben Eigenwerte wie G[40]. Sei nun $x \in \mathbb{C}^n$, $x^H x = 1$[41], ein Eigenvektor von G_1 zum Eigenwert λ, dann heißt dies, daß

$$-L_1^T x = \lambda \ (I + L_1)\, x$$

und somit

$$-\underbrace{x^H L_1^T x}_{=(L_1 x)^H x = (x^H L_1 x)^H = \overline{x^H L_1 x}} = \lambda \left(1 + x^H L_1 x\right).$$

[39] Sonst wäre $e_j^T A e_j \leq 0$ für ein j.

[40] Ist y ein Eigenvektor von G zum Eigenwert λ, dann ist $D^{1/2} y$ ein Eigenvektor von G_1 zum Eigenwert λ und umgekehrt.

[41] $x^H = [\overline{x}_j \ : \ j = 1, \ldots, n]^T$

```
%%  Jacobi.m (Numerik 1)
%%  ----------------------------------------------------------
%%  Jacobi-Verfahren
%%  Eingabe:
%%    A    Matrix
%%    b    rechte Seite
%%    tol  Genauigkeit

function x = Jacobi( A,b,tol )
  n = length( A );
  x = zeros( n,1 );
  nrm = norm( A*x - b );

  while ( nrm  > tol )
    disp( nrm );
    x = JacobiIt( A,x,b );
    nrm = norm( A*x - b );
  end
%endfunction
```

Programm 6.2 `Jacobi.m`: Das Jacobi–Verfahren.

```
%%  GaussSeidelIt.m (Numerik 1)
%%  ---------------------------------------------------
%%  Gauss-Seidel-Iterationsschritt mit Ueberschreiben von x
%%  Eingabe:
%%    A    Matrix
%%    b    rechte Seite
%%    x    Naeherung fuer Loesung

function y = GaussSeidelIt( A,x,b )
  n = length( A );

  for j = 1:n
    z = b ( j ) - A( j,1:j-1 ) * x( 1:j-1 ) - A( j,j+1:n ) * x( j+1:n );
    x( j ) = z / A( j,j ); %% x statt y !!!
  end
  y = x;
%endfunction
```

Programm 6.3 `GaussSeidelIt.m`: Eine Gauß–Seidel–Iteration nach (6.11).

```
%%  GaussSeidel.m (Numerik 1)
%%  ---------------------------------------------------
%%  Gauss-Seidel-Verfahren
%%  Eingabe:
%%    A    Matrix
%%    b    rechte Seite
%%    tol  Genauigkeit

function x = GaussSeidel( A,b,tol )
  n = length( A );
  x = zeros( n,1 );
  nrm = norm( A*x - b );

  while ( nrm  > tol )
    disp( nrm );
    x = GaussSeidelIt( A,x,b );
    nrm = norm( A*x - b );
  end
%endfunction
```

Programm 6.4 `GaussSeidel.m`: Das Gauß–Seidel–Verfahren.

Schreiben wir $x^H L_1 x = \alpha + i\beta$, dann ist

$$|\lambda|^2 = \left|\frac{-\alpha + i\beta}{1 + \alpha + i\beta}\right|^2 = \frac{\alpha^2 + \beta^2}{1 + 2\alpha + \alpha^2 + \beta^2}. \tag{6.13}$$

Auf der anderen Seite ist A und damit auch $D^{-1/2} A D^{-1/2} = I + L_1 + L_1^T$ symmetrisch und positiv definit[42], also

$$0 < x^H \left(I + L_1 + L_1^T\right) x = 1 + \underbrace{x^H L_1 x}_{=\alpha+i\beta} + \underbrace{x^H L_1^T x}_{=\alpha-i\beta} = 1 + 2\alpha,$$

was, in (6.13) eingesetzt, $|\lambda| < 1$ liefert. □

Definition 6.11 *Eine Matrix $A \in \mathbb{R}^{n\times n}$ heißt* (zeilenweise) diagonaldominant*, wenn die Diagonalelemente den "Rest" der Zeile* im Abosultbetrag *dominieren, wenn also*

$$a_{jj} \geq \sum_{k\neq j} |a_{jk}|, \qquad j = 1, \ldots, n,$$

gilt. Diagonaldominante Matrizen haben insbesondere nichtnegative Diagonaleinträge.

Satz 6.12 *(Diagonaldominanz und Konvergenz)*
Ist $A \in \mathbb{R}^{n\times n}$ zeilenweise strikt diagonaldominant, so konvergieren das Jacobi- und das Gauß–Seidel–Verfahren.

Beweis: Für die Zerlegung aus dem Jacobi–Verfahren ($B = \operatorname{diag} A$) gilt

$$\left\|I - B^{-1}A\right\|_\infty = \max_{j=1,\ldots,n} \frac{1}{|a_{jj}|} \sum_{k\neq j} |a_{jk}|$$

und im Falle strikt diagonaldominanter Matrizen ist somit

$$\rho\left(I - B^{-1}A\right) \leq \left\|I - B^{-1}A\right\|_\infty < 1.$$

Für Gauß–Seidel müssen wir uns ein bißchen mehr anstrengen. Hierzu schreiben wir

$$A = L + D + U, \qquad \ell_{jj} = u_{jj} = 0,$$

wobei $d_{jj} \neq 0$ ist, und setzen $L_1 = -D^{-1}L$, $U_1 = -D^{-1}U$. Die zu betrachtende Matrix ist

$$\begin{aligned} I - B^{-1}A &= I - (L+D)^{-1}(L+D+U) = I - (-DL_1 + D)^{-1}(L+D+U) \\ &= I - (I-L_1)^{-1} D^{-1}(L+D+U) = I - (I-L_1)^{-1}(I - L_1 - U_1) \\ &= I - I + (I-L_1)^{-1} U_1 = (I-L_1)^{-1} U_1 =: H, \end{aligned}$$

[42] Zur Erinnerung an die Lineare Algebra: Bei einer *komplexen* Matrix gehört es zur Definition von positiver Definitheit, daß die Matrix hermitesch ist. Bei rellen Matrizen fordert man es oder eben auch nicht ...

außerdem schreiben wir $J := L_1 + U_1 = I - D^{-1}A$. Da $\|1_n\|_\infty = 1$, ist (wg. der strikten Zeilendiagonaldominanz)

$$|J|\, 1_n \leq \|J\|_\infty 1_n < 1_n$$

und da $|J| = |L_1| + |U_1|$, ist

$$|U_1|\ 1_n = (|J| - |L_1|)\ 1_n \leq (\|J\|_\infty I - |L_1|)\, 1_n. \tag{6.14}$$

L_1 und $|L_1|$ sind strikte untere Dreiecksmatrizen und damit *nilpotent*, das heißt,

$$L_1^n = |L_1|^n = 0 \qquad \Longrightarrow \qquad \left\{ \begin{array}{c} (I - L_1)^{-1} = I + L_1 + \cdots + L_1^{n-1} \\ (I - |L_1|)^{-1} = I + |L_1| + \cdots + |L_1|^{n-1} \end{array} \right.$$

und damit ist

$$\left|(I - L_1)^{-1}\right| = \left|I + L_1 + \cdots + L_1^{n-1}\right| \leq I + |L_1| + \cdots + |L_1|^{n-1} = (I - |L_1|)^{-1}. \tag{6.15}$$

Mit (6.14) und (6.15) ist demnach

$$\begin{array}{rcl} |H|\, 1_n & \leq & \left|(I - L_1)^{-1}\right|\ |U_1|\ 1_n \leq (I - |L_1|)^{-1} (\|J\|_\infty I - |L_1|)\ 1_n \\ & = & (I - |L_1|)^{-1} (I - |L_1| + (\|J\|_\infty - 1)I)\ 1_n \\ & = & 1_n + \underbrace{(\|J\|_\infty - 1)}_{<0} \underbrace{(I - |L_1|)^{-1}}_{\geq I} 1_n \\ & \leq & (I + (\|J\|_\infty - 1)I)\ 1_n \leq \|J\|_\infty\ 1_n < 1_n. \end{array}$$

Also ist $|H|\ 1_n < 1_n$, also alle Zeilensummen von $|H|$ kleiner als 1 und damit ist $\|H\|_\infty < 1$, also auch $\rho(H) < 1$. □

6.3 Relaxationsverfahren

Offensichtlich hängt die Qualität eines Iterationsverfahrens von der Wahl der Matrix B ab, die ja auch die Konvergenzgeschwindigkeit, also $\rho\,(I - B^{-1}A)$ beeinflußt. Die *Relaxationsverfahren* sind Variationen der Gauß–Seidel–Iteration, bei denen eine Familie $B(\omega)$ betrachtet wird, wobei man $\omega > 0$ als den *Relaxationsparameter* bezeichnet. Genauer: man setzt, für $A = L + D + U$

$$B(\omega) = \frac{1}{\omega}\,(D + \omega L)\,,$$

also

$$\frac{1}{\omega}\,(D + \omega L)\ x^{(j+1)} = b - \frac{1}{\omega}\,((1 - \omega)D + \omega U)\ x^{(j)}, \tag{6.16}$$

das heißt,

$$x_k^{(j+1)} = \omega \left(b_k - \sum_{r=0}^{k-1} a_{kr} x_r^{(j+1)} - \sum_{r=k+1}^{n} a_{kr} x_r^{(j)} \right) / a_{kk} + (1 - \omega)\, x_k^{(j)}, \tag{6.17}$$

und betrachtet den Spektralradius von $H(\omega) := I - B(\omega)^{-1}A$.

```
%%  RelaxIt.m (Numerik 1)
%%  ---------------------------------------------------
%%  Gauss-Seidel-Relxation-Iterationsschritt
%%     mit Ueberschreiben von x
%%  Eingabe:
%%    A    Matrix
%%    x    Naeherung fuer Loesung
%%    b    rechte Seite
%%    r    Relaxationsparameter

function y = RelaxIt( A,x,b,r )
  n = length( A );

  for j = 1:n
    z = b ( j ) - A( j,1:j-1 ) * x( 1:j-1 ) - A( j,j+1:n ) * x( j+1:n );
    x( j ) = r * z / A( j,j ) + (1-r) * x( j );
  end
%endfunction
```

Programm 6.5 `RelaxIt.m`: Iterationsschritt (6.17) des Relaxationsverfahrens.

Bemerkung 6.13 *(Relaxationsverfahren)*

1. *Ist* $\omega < 1$ *spricht man von* Unterrelaxation*, andernfalls von* Überrelaxation*.*
2. *Für* $\omega = 1$ *erhält man das Gauß–Seidel–Verfahren.*
3. *Für bestimmte Matrizen*[43]*, für die das Gauß–Seidel–Verfahren konvergiert, ist die Funktion* $\omega \mapsto \rho\left(H(\omega)\right)$ *monoton fallend auf einem Intervall* $[1, \bar{\omega}]$*, hier bringt also* SOR *(“***S***uccessive* **O***ver***R***elaxation” wirklich etwas.*
4. *Man kann zeigen*[44]*, daß*
$$\rho\left(H(\omega)\right) \geq |\omega - 1|\,,$$
also macht $\omega < 0$ *und* $\omega > 2$ *keinen Sinn.*

6.4 Konjugierte Gradienten

Das Verfahren der konjugierten Gradienten (*“conjugate gradients”*) ist auf symmetrische, positiv definite Matrizen zugeschnitten. Ist $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit, dann

[43] Im wesentlichen sollen $L, U \geq 0$ sein.
[44] Das Resultat geht auf Kahan (wieder mal) zurück.

```
%%  Relax.m (Numerik 1)
%%  -----------------------------------------------------
%%  Gauss-Seidel-Relaxations-Verfahren
%%  Eingabe:
%%    A    Matrix
%%    b    rechte Seite
%%    tol  Genauigkeit
%%    r    Relaxationsparameter

function x = Relax( A,b,tol,r )
  n = length( A );
  x = zeros( n,1 );
  nrm = norm( A*x - b );

  while ( nrm  > tol )
    disp( nrm );
    x = GaussSeidelIt( A,x,b,r );
    nrm = norm( A*x - b );
  end
%endfunction
```

Programm 6.6 `Relax.m`: Gauß–Seidel–Relaxationsverfahren.

gibt es nach Satz 5.4 eine invertierbare untere Dreiecksmatrix $G \in \mathbb{R}^{n\times n}$, so daß $A = GG^T$ und damit ist auch $A^{-1} = G^{-T}G^{-1}$ symmetrisch und positiv definit. Wir betrachten die Funktion $F : \mathbb{R}^n \to \mathbb{R}$, definiert als

$$F(x) = \frac{1}{2}\left(Ax - b\right)^T A^{-1}\left(Ax - b\right). \tag{6.18}$$

Da A^{-1} positiv definit ist, ist

$$F(x) \geq 0 \qquad \text{und} \qquad F(x) = 0 \quad \Longleftrightarrow \quad Ax = b,$$

das heißt, man ersetzt das Lösen eine linearen Gleichungssystems durch das Auffinden des globalen Minimums einer quadratischen Funktion. Da

$$F(x) = \frac{1}{2}x^T Ax - x^T b + \frac{1}{2}b^T A^{-1}b, \qquad x \in \mathbb{R}^n,$$

ist das x, das F minimiert genau dasselbe wie das x, das den Ausdruck $x \mapsto \frac{1}{2}x^T Ax - x^T b$ minimiert. Aus diesem Grund werden wir den konstanten Term $\frac{1}{2}b^T A^{-1}b$ bei F mal mitführen und mal weglassen, je nachdem, was gerade nützlicher ist[45]. Das Ziel wird es sein, eine Folge $x^{(j)}$, $j \in \mathbb{N}$, zu konstruieren, so daß

$$F\left(x^{(j+1)}\right) < F\left(x^{(j)}\right) \qquad \text{und} \qquad \lim_{j\to\infty} F\left(x^{(j)}\right) = 0,$$

und zwar durch Lösen *eindimensionaler* Optimierungsprobleme.

Lemma 6.14 *Für $x, y \in \mathbb{R}^n$ nimmt die Funktion*

$$f(t) = F\left(x + ty\right)$$

ihr Minimum an der Stelle

$$t = \frac{y^T\left(b - Ax\right)}{y^T Ay} \tag{6.19}$$

an.

Beweis: Die Funktion

$$f(t) = F(x) + \frac{t^2}{2}\underbrace{y^T Ay}_{>0} + t\left(x^T Ay - y^T b\right) \tag{6.20}$$

hat genau ein Minimum und

$$0 = f'(t) = t\, y^T Ay - \left(y^T b - x^T Ay\right) = t\, y^T Ay - y^T\left(b - Ax\right)$$

liefert (6.19). □

Das liefert uns bereit das allgemeine Iterationsverfahren:

[45] Oder dem Zeitgeist entspricht.

```
%%  SteepDesc.m (Numerik 1)
%%  -----------------------------------------------------
%%  Steilster Abstieg, N Iterationen
%%  Eingabe:
%%    A    Matrix (SPD)
%%    b    rechte Seite
%%    N    % Iterationen

function x = SteepDesc( A,b,N )
  n = length( A );
  x = zeros( n,1 );
  r = b - A*x;

  for j = 1:N
    disp( norm( r ) );
    x = x + (r' * r) / (r' * A * r) * r;
    r = b - A*x;
  end
%endfunction
```

Programm 6.7 `SteepDesc.m`: Methode des steilsten Abstiegs.

1. Setze $r_j := b - Ax^{(j)}$ (Residuum!)
2. Ist $r_j = 0$, dann ist $x^{(j)}$ eine Lösung von $Ax = b$, andernfalls wähle $y \in \mathbb{R}^n \setminus \{0\}$ und setze
$$x^{(j+1)} = x^{(j)} + \frac{r_j^T y}{y^T A y} y.$$

Ist nicht gerade $y \perp r_j$, dann ist tatsächlich $F\left(x^{(j+1)}\right) < F\left(x^{(j)}\right)$. Die erste Idee wäre sicherlich, y als Richtung des *steilsten Abstiegs* , also

$$y = -DF(x) = -Ax + b = r_j$$

(Gradient!) zu wählen, was das Iterationsverfahren `SteepDesc.m` liefert. Leider kann es sehr schnell passieren, daß dieses Verfahren beliebig langsam, das heißt numerisch gar nicht, konvergiert.

Lemma 6.15 *Es seien $\lambda_1 \leq \ldots \leq \lambda_n$ die Eigenwerte von A. Dann ist*

$$F\left(x^{(j+1)}\right) \leq \left(1 - \frac{\lambda_1}{\lambda_n}\right) F\left(x^{(j)}\right), \qquad j \in \mathbb{N}_0. \tag{6.21}$$

Beweis: Für $j \in \mathbb{N}_0$ ist

$$\begin{aligned} F\left(x^{(j+1)}\right) &= F\left(x^{(j)}\right) + \frac{1}{2}\left(\frac{r_j^T r_j}{r_j^T A r_j}\right)^2 r_j^T A r_j + \frac{r_j^T r_j}{r_j^T A r_j} r_j^T \underbrace{\left(Ax^{(j)} - b\right)}_{=-r_j} \\ &= F\left(x^{(j)}\right) - \frac{1}{2}\frac{\left(r_j^T r_j\right)^2}{r_j^T A r_j} = F\left(x^{(j)}\right) - \frac{1}{2}\frac{\|r_j\|_2^4}{r_j^T A r_j} \le F\left(x^{(j)}\right) - \frac{1}{2}\frac{\|r_j\|_2^4}{\lambda_n \|r_j\|_2^2} \\ &= F\left(x^{(j)}\right) - \frac{1}{2}\frac{r_j^T r_j}{\lambda_n} = \frac{1}{2} r_j^T \left(A^{-1} - \frac{1}{\lambda_n} I\right) r_j \le \frac{1}{2} r_j^T \left(A^{-1} - \frac{\lambda_1}{\lambda_n} A^{-1}\right) r_j \\ &= \left(1 - \frac{\lambda_1}{\lambda_n}\right) F\left(x^{(j)}\right). \end{aligned}$$

Der letzte Schritt ist wegen $x^T A^{-1} x \le \lambda_1^{-1} x^T x$ richtig[46]. □

Übung 6.1 Zeigen Sie: Ist $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit und ist λ der größte Eigenwert von A, dann ist

$$x^T A x \le \lambda \|x\|_2^2, \qquad x \in \mathbb{R}^n.$$

Dieses Verhalten kann man auch praktisch in `Matlab` sehen: man startet mit einer zufälligen $n\times n$–Matrix `A = rand( n )` und startet die Methode des steilsten Abstiegs mit einer Matrix der Form

```
A' * A + t eyes( n )
```

für verschiedene Werte von `t`> 0. Je größer `t` ist, desto besser wird die Methode funktionieren und konvergieren, für `t` $= 0$ hingegen kann man normalerweise nicht mehr von Konvergenz sprechen (die Matrix wird fast singulär). Warum das so ist und was die geometrische Interpretation ist, sieht man einfach am folgenden Beispiel.

Beispiel 6.16 *Wir betrachten*

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 10^{-3} \end{bmatrix} \qquad \textit{und} \qquad b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Die Lösung ist natürlich $x = [1, 1000]^T$. *Für Vektoren* $x = [x_1, x_2]^T$ *mit moderatem* x_2 *ist dann*

$$r := -DF(x) = b - Ax = \begin{bmatrix} 1 - x_1 \\ 1 - 10^{-3} x_2 \end{bmatrix} \approx \begin{bmatrix} 1 - x_1 \\ 1 \end{bmatrix}$$

und

$$\alpha := \frac{r^T r}{r^T A r} \sim \frac{1 + (1 - x_1)^2}{(1 - x_1)^2}.$$

[46] Der größte Eigenwert von A^{-1} ist λ_1^{-1}.

Ist nun $x_1 \sim 0$ oder $x_1 \sim 2$, dann ist $\alpha \sim 2$ und damit ist

$$x + \alpha r \sim \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 2 \begin{bmatrix} 1 - x_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 - x_1 \\ x_2 + 2 \end{bmatrix}$$

Starten wir also mit $x^{(0)} = 0$, so werden die Werte $x_1^{(j)}$ anfangs zwischen 0 und 2 pendeln[47] *und die Werte $x_2^{(j)} \sim 2j$ sein. Kommt dann das Verfahren richtig "in Fahrt" (also gegen Ende des Verfahrens), dann wird auch die Konvergenz deutlich schneller.*
Geometrisch bedeutet dies, daß sich das Verfahren an "flachgedrückten" Ellipsen entlanghangelt, siehe Abb. 6.1.

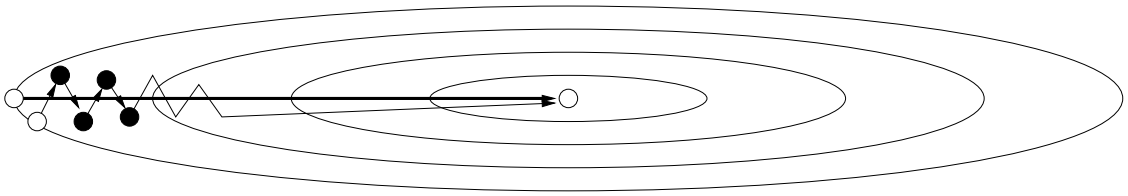


Abbildung 6.1: Verfahren des steilsten Abstiegs für Beispiel 6.16 (nicht maßstabsgetreu). Beachte: mit dem "richtigen" Startwert (nicht ganz zufällig ein Eigenvektor von A) würde das Verfahren nach *einem* Schritt erfolgreich terminieren.

Wir brauchen also etwas anderes. Die Idee der *konjugierten Gradienten* besteht nun darin, sukzessive Richtungen $v_1, v_2, \ldots \in \mathbb{R}^n$ und Punkte

$$x^{(j)} \in V_j := \text{span}\,\{v_1, \ldots, v_j\}\,, \qquad x^{(0)} = 0$$

zu konstruieren, so daß

$$F\left(x^{(j)}\right) = \min\left\{F(v) \;:\; v \in \text{span}\,\{v_1, \ldots, v_j\}\right\} \tag{6.22}$$

und dies auch noch durch Lösung von *eindimensionalen* Optimierungsproblemen wie in Lemma 6.14. Kaum zu glauben, aber das funktioniert tatsächlich, wenn nur die Vektoren v_j, $j \in \mathbb{N}$, richtig gewählt werden.

Definition 6.17 *Sei $A \in \mathbb{R}^{n\times n}$ und $V \subseteq \mathbb{R}^n$. Ein Vektor $x \in \mathbb{R}^n$ heißt A–*konjugiert zu V*, falls*

$$x^T A V = 0, \qquad \textit{d.h.} \qquad x^T A v = 0, \quad v \in V.$$

Lemma 6.18 *Ist v_{j+1} A–konjugiert zu V_j und erfüllt $x^{(j)}$ die Bedingung (6.22), dann erfüllt*

$$x^{(j+1)} = x^{(j)} + \frac{r_j^T v_{j+1}}{v_{j+1}^T A v_{j+1}} v_{j+1} \tag{6.23}$$

ebenfalls die Bedingung (6.22)[48].

[47] In Wirklichkeit sind sie stets etwas größer als 0 und etwas kleiner als 2 und dieses "etwas" wächst.
[48] Natürlich mit $j + 1$ anstelle von j.

Beweis: Da

$$F\left(x^{(j)}+tv_{j+1}\right)=F\left(x^{(j)}\right)+\frac{t^2}{2}v_{j+1}^TAv_{j+1}+t\underbrace{v_{j+1}^TAx^{(j)}}_{=0}-tv_{j+1}^Tb=F\left(x^{(j)}\right)+f(t)$$

ist das Optimierungsproblem in eines auf V_j und eines bezüglich v_{j+1} entkoppelt. Nach (6.22) minimiert $x^{(j)}$ den ersten Teil und $f'(t)=0$ ergibt den Anteil bezüglich v_{j+1}, nämlich

$$t=\frac{b^Tv_{j+1}}{v_{j+1}^TAv_{j+1}}=\frac{r_j^Tv_{j+1}}{v_{j+1}^TAv_{j+1}},\qquad\text{da}\qquad\left(Ax^{(j)}\right)^Tv_{j+1}=v_{j+1}^TAx^{(j)}=0.$$

□

Die so konstruierten $x^{(j)}$, bzw. die davon erzeugten Residuen haben eine schöne Eigenschaft: es ist nämlich für $k\le j$

$$r_j^T\,v_k=\left(b-A\sum_{\ell=1}^{j}\frac{b^Tv_\ell}{v_\ell^TAv_\ell}\,v_\ell\right)^T v_k=b^Tv_k-\sum_{\ell=1}^{j}\frac{b^Tv_\ell}{v_\ell^TA\,v_\ell}\,\underbrace{v_\ell^TA\,v_k}_{=\delta_{k,\ell}}=b^Tv_k-b^Tv_k=0,$$

also

$$r_j^T\,V_j=0,\qquad j\in\mathbb{N}. \tag{6.24}$$

Außerdem erhalten wir die Rekursionsformel

$$r_{j+1}=b-Ax^{(j+1)}=b-A\left(x^{(j)}+\frac{r_j^Tv_{j+1}}{v_{j+1}^TAv_{j+1}}v_{j+1}\right)=r_j-\frac{r_j^Tv_{j+1}}{v_{j+1}^TAv_{j+1}}Av_{j+1}, \tag{6.25}$$

Für ein funktionierendes Verfahren müssen wir jetzt also "nur" noch einen konjugierten Vektor finden. Das geht aber.

Proposition 6.19 *Es seien $v_1,\dots,v_j$ A–konjugierte Vektoren, das heißt, $v_kAv_\ell=0$, $1\le k\ne\ell\le j$. Dann ist der Vektor*

$$v_{j+1}=r_j-\frac{v_j^TAr_j}{v_j^TAv_j}v_j \tag{6.26}$$

A–konjugiert zu V_j.

Zuerst halten wir fest, daß diese Iteration "normalerweise" kein $v_{j+1}=0$ erzeugen kann.

Lemma 6.20 *In (6.26) ist*

$$v_{j+1}\in V_j\qquad\Longleftrightarrow\qquad v_{j+1}=0\qquad\Longleftrightarrow\qquad r_j=0.$$

Beweis: Ist $v_{j+1} \in V_j$ oder $v_{j+1} = 0$, dann ist, mit (6.24),

$$0 = r_j^T v_{j+1} = r_j^T r_j - \frac{v_j^T A r_j}{v_j^T A v_j} \underbrace{r_j^T v_j}_{=0} = \|r_j\|_2^2 ;$$

die Umkehrungen sind trivial. □

Um Proposition 6.19 beweisen zu können, müssen wir uns erst einmal die Struktur der so erzeugten Räume V_j ansehen.

Lemma 6.21 *Sind die durch (6.26) erzeugten Vektoren $v_1, \ldots, v_j \neq 0$, dann erfüllen die Räume V_j*

$$V_j = \text{span}\,\{r_k \;:\; k = 0, \ldots, j-1\} = \text{span}\,\left\{A^k b \;:\; k = 0, \ldots, j-1\right\}. \tag{6.27}$$

Beweis: Induktion über j. Für $j = 1$ ist, nach (6.26)

$$v_1 = r_0 = b,$$

also ist (6.27) trivialerweise richtig.
Sei also (6.27) für ein $j \geq 1$ bewiesen, dann sagt uns die Rekursionsformel (6.25), daß

$$r_j = \underbrace{r_{j-1}}_{\in V_j} - \underbrace{\frac{b^T v_j}{v_j^T A\, v_j}\, A\, v_j}_{\in A\, V_j} \quad \in V_j + A\, V_j,$$

also, nach (6.26), auch

$$v_{j+1} \in V_j + A\, V_j$$

und nach der Induktionsannahme ist damit

$$V_{j+1} \subseteq \left\{A^k b \;:\; k = 0, \ldots, j\right\}.$$

Die Gleichheit ergibt sich aus Dimensionsüberlegungen, denn nach der Voraussetzung und Lemma 6.20 sind $v_1, \ldots, v_{j+1}$ linear unabhängig. □

Beweis von Proposition 6.19: Es ist

$$v_{j+1}^T A\, v_k = r_j^T A\, v_k - \frac{v_j^T A r_j}{v_j^T A v_j}\, v_j^T A\, v_k = \begin{cases} 0 & k = j, \\ r_j^T A\, v_k & k < j, \end{cases}$$

und nach Lemma 6.21 ist $A\, v_k \in V_{k+1} \subseteq V_j$ (da $k < j$) und somit ist, nach (6.24) auch

$$r_j^T A\, v_k = 0, \qquad k = 1, \ldots, j-1.$$

□

Damit können wir unser Verfahren für konjugierte Gradienten zusammensetzen.

```
%%  ConjGrad.m (Numerik I)
%%  ------------------------------------------------------
%%  Conjugate Gradients
%%  Eingabe:
%%    A    Matrix (SPD)
%%    b    rechte Seite
%%    N    % Iterationen

function x = ConjGrad( A,b,N )
  n = length( A );
  x = zeros( n,1 );
  r = b;
  v = b;

  for j = 1:N
    disp( norm( r ) );
    x = x + ( r' * v ) / ( v' * A * v ) * v;
    r = b - A*x;
    v = r - ( v' * A * r ) / ( v' * A * v ) * v;
  end
%endfunction
```

Programm 6.8 `ConjGrad.m`: Die CG–Methode (**C**onjugate **G**radients), Version 0.

1. Setze $x^{(0)} = 0$, $r_0 = b$, $v_1 = b$.
2. Für $j \in \mathbb{N}$ berechne

$$\begin{aligned} x^{(j)} &:= x^{(j-1)} + \frac{r_{j-1}^T v_j}{v_j^T A\, v_j} v_j \\ r_j &:= b - A\, x^{(j)} \\ v_{j+1} &:= r_j - \frac{v_j^T A r_j}{v_j^T A\, v_j} v_j \end{aligned}$$

Satz 6.22 *In exakter Rechnung terminiert das CG–Verfahren spätestens nach n Schritten mit einer Lösung von $Ax = b$.*

Beweis: Jedes v_j, das in (6.26) ist von V_{j-1} linear unabhängig oder 0. Da es im $\mathbb{R}^n$ höchstens n linear unabhängige Vektoren geben kann, ist also spätestens $v_{n+1} = 0$, also, nach Lemma 6.20, war bereits $r_n = 0$ und damit $x^{(n)}$ eine Lösung von $Ax = b$. □

```
%%  ConjGrad.m (Numerik I)
%%  ------------------------------------------------------
%%  Conjugate Gradients verwende b (siehe Skript)
%%  Eingabe:
%%    A    Matrix (SPD)
%%    b    rechte Seite
%%    N    % Iterationen

function x = ConjGrad1( A,b,N )
  n = length( A );
  x = zeros( n,1 );
  r = b;
  v = b;

  for j = 1:N
    disp( norm( r ) );
    x = x + ( b' * v ) / ( v' * A * v ) * v;
    r = b - A*x;
    v = r - ( v' * A * r ) / ( v' * A * v ) * v;
  end
%endfunction
```

Programm 6.9 `ConjGrad1.m`: Theoretisch äquivalentes CG–Verfahren.

Bemerkung 6.23 *(Konjugierte Gradienten)*

1. *Gedacht war das CG–Verfahren ursprünglich als ein* exaktes *Verfahren. Wegen der allgegenwärtigen Rundungsfehler berechnet es aber keine exakte Lösung. Deswegen iteriert man einfach auf gut Glück weiter, bis die gewünschte Toleranz erreicht wird.*

2. *Die Verwendung von r_j^T anstelle von b^T stabilisiert die Rechnung entscheidend: Zwar sind beide Werte theoretisch gleichwertig, aber bei nicht exakter Rechnung ergeben sich erhebliche Differenzen, siehe* `ConjGrad1.m`.

3. *Durch Umformungen und Einsetzen der Rekursionsformeln kommt man auf die einfacheren Iterationsvorschriften*

$$\begin{aligned} x^{(j)} &:= x^{(j-1)} + \frac{r_{j-1}^T r_{j-1}}{v_j^T A\, v_j} v_j \\ r_j &:= b - A\, x^{(j)} \end{aligned}$$

```
%%  ConjGrad2.m (Numerik I)
%%  ------------------------------------------------------
%%  Conjugate Gradients, Abbruch nach Toleranz
%%  Eingabe:
%%    A    Matrix (SPD)
%%    b    rechte Seite
%%    tol  vorgegebene Toleranz

function x = ConjGrad2( A,b,tol )
  n = length( A );
  x = zeros( n,1 );
  r = b;
  v = b;

  rn = r'*r;

  while ( rn > tol )
    disp( rn );
    x = x + rn / ( v' * A * v ) * v;
    r = b - A*x;
    v = r + ( r' * r ) / rn * v;

    rn = r' * r;
  end
%endfunction
```

Programm 6.10 `ConjGrad2.m`: Etwas schnelleres CG–Verfahren.

$$v_{j+1} \;:=\; r_j + \frac{r_j^T r_j}{r_{j-1}^T r_{j-1}} v_j$$

was zu `ConjGrad2.m` *führt.*

And now for something completely different ...

Monty Python's Flying Circus

7 Polynome

Bei vielen praktischen Problemen (z.B. CAD) ist es wichtig, Funktionen näherungsweise durch "einfache" Funktionen (die mit endlich viel Information dargestellt werden können) zu repräsentieren oder zu approximieren. Die beiden wesentlichen Anwendungen sind

Modellierung: Man versucht, durch Verändern von Parametern eine Kurve (oder Fläche) zu erhalten, die (zumindest näherungsweise) die gewünschte Form hat.

Rekonstruktion: Aus endlicher Information (z.B. Abtastwerte) soll eine Kurve (oder Fläche) gewonnen werden, die diese Information reproduziert (z.B. interpoliert).

Die einfachsten und ältesten Funktionen für diese Zwecke sind die *Polynome*.

Definition 7.1 *Eine Funktion der Form*

$$p(x) = \sum_{j \in \mathbb{N}_0} a_j x^j, \qquad a_j \in \mathbb{K},$$

heißt Polynom über dem Körper $\mathbb{K}$*, falls*

$$\deg p := \max \{j \in \mathbb{N}_0 \ : \ a_j \neq 0\} < \infty.$$

Die Zahl $\deg p$ *nennt man den* Grad *von* p*, mit*

$$\Pi_n = \{p : p \textit{ Polynom}, \deg p \leq n\}$$

bezeichnet man den Vektorraum aller Polynome vom Grad höchstens n *und mit* $\Pi = \mathbb{K}[x]$ *den Vektorraum aller Polynome.*

Bemerkung 7.2 *(Polynome)*

1. *Die Vektorräume* Π_n*,* $n \in \mathbb{N}$*, und* Π *sind* $\mathbb{K}$*–Vektorräume, also abgeschlossen unter Multiplikation mit* $\mathbb{K}$*.*

2. *Es ist*

$$\dim \Pi_n = n + 1, \qquad n \in \mathbb{N}_0$$

und die Monome $\{1, x, \ldots, x^n\}$ *bilden eine Basis von* Π_n*.*

3. Π *ist sogar eine* Algebra*: Das Produkt* pq *zweier Polynome* $p, q \in \Pi$ *ist wieder ein Polynom.*

Ein großer Vorteil von Polynomen ist, daß sie in den stetigen Funktionen auf jedem kompakten Intervall dicht liegen, das heißt, daß wir mit ihnen jede stetige Funktion beliebig genau annähern können; das ist der berühmte Satz von Weierstrass (1885), siehe [49].

Satz 7.3 *Es sei* $I \subset \mathbb{R}$ *ein kompaktes Intervall und* $f \in C(I)$*. Dann gibt es zu jedem* $\varepsilon > 0$ *ein* $p \in \Pi$*, so daß*

$$\|f - p\|_I := \max_{x \in I} |f(x) - p(x)| < \varepsilon. \tag{7.1}$$

Beweis: Ohne Einschränkung können wir annehmen, daß $I = [0, 1]$. Dann können wir eine Folge von approximierenden Polynomen *explizit* angeben, nämlich die *Bernsteinpolynome*[49]

$$(B_n f)(x) = \sum_{j=0}^{n} f(j/n) \underbrace{\binom{n}{j} x^j (1-x)^{n-j}}_{=:B_j^n(x)} \quad \in \Pi_n. \tag{7.2}$$

Nun ist

$$\sum_{j=0}^{n} B_j^n(x) = \sum_{j=0}^{n} \binom{n}{j} x^j (1-x)^{n-j} = (x + 1 - x)^n = 1,$$

sowie

$$\begin{aligned} \sum_{j=0}^{n} \frac{j}{n} B_j^n(x) &= \sum_{j=0}^{n} \frac{j}{n} \frac{n!}{j!(n-j)!} x^j (1-x)^{n-j} = \sum_{j=1}^{n} \frac{(n-1)!}{(j-1)!(n-j)!} x^j (1-x)^{n-j} \\ &= x \sum_{j=1}^{n} \frac{(n-1)!}{(j-1)!(n-j)!} x^{j-1} (1-x)^{n-j} = x \sum_{j=0}^{n-1} B_j^{n-1}(x) = x \end{aligned}$$

und

$$\begin{aligned} \sum_{j=0}^{n} \frac{j(j-1)}{n^2} B_j^n(x) &= \sum_{j=0}^{n} \frac{j(j-1)}{n^2} \frac{n!}{j!(n-j)!} x^j (1-x)^{n-j} \\ &= \frac{n-1}{n} \sum_{j=2}^{n} \frac{(n-2)!}{(j-2)!(n-j)!} x^j (1-x)^{n-j} = \frac{n-1}{n} x^2 \sum_{j=0}^{n-2} B_j^{n-2}(x) \\ &= \frac{n-1}{n} x^2. \end{aligned}$$

[49] Der Name kommt daher, daß der Beweis im Original von S. Bernstein [3] stammt, dem übrigens die wahrscheinlichkeitstheoretische Idee durchaus klar war.

Mit diesen drei Identitäten erhalten wir, daß für jedes $x \in [0,1]$

$$\begin{aligned}\sum_{j=0}^{n}\left(\frac{j}{n}-x\right)^2 B_j^n(x) &= \sum_{j=0}^{n}\left(\frac{j^2}{n^2}-2\frac{j}{n}x+x^2\right)B_j^n(x)\\ &= \sum_{j=0}^{n}\left(\frac{j(j-1)}{n^2}+\frac{j}{n^2}-2\frac{j}{n}x+x^2\right)B_j^n(x)\\ &= \frac{n-1}{n}x^2+\frac{1}{n}x-2x^2+x^2=\frac{1}{n}\,x(1-x)\end{aligned}$$

Diese Formel erlaubt es uns nun zu zeigen, daß die Basispolynome recht gut lokalisiert sind: für $\delta > 0$ und $x \in [0,1]$ ist

$$\sum_{\left|\frac{j}{n}-x\right|\geq\delta} B_j^n(x) \leq \frac{1}{\delta^2}\sum_{j=0}^{n}\left(\frac{j}{n}-x\right)^2 B_j^n(x) = \frac{x(1-x)}{n\delta^2} \leq \frac{1}{4n\delta^2}. \tag{7.3}$$

Das war's dann auch schon fast! Sei nun $f \in C[0,1]$, dann ist f ja nicht nur stetig, sondern sogar *gleichmäßig stetig*, das heißt, für alle $\varepsilon > 0$ gibt es ein $\delta > 0$ so daß

$$|x-y| < \delta \qquad \Longrightarrow \qquad |f(x)-f(y)| < \frac{\varepsilon}{2}.$$

Sei also $\varepsilon > 0$ vorgegeben und $\delta > 0$ passend gewählt. Dann ist, für beliebiges $x \in [0,1]$,

$$\begin{aligned}|f(x)-B_nf(x)| &= \left|\sum_{j=0}^{n}(f(x)-f(j/n))\;B_j^n(x)\right| \leq \sum_{j=0}^{n}|f(x)-f(j/n)|\;B_j^n(x)\\ &= \sum_{\left|\frac{j}{n}-x\right|<\delta}\underbrace{|f(x)-f(j/n)|}_{\leq\varepsilon/2}\;B_j^n(x)+\sum_{\left|\frac{j}{n}-x\right|\geq\delta}\underbrace{|f(x)-f(j/n)|}_{\leq 2\|f\|_I}\;B_j^n(x)\\ &\leq \frac{\varepsilon}{2}\sum_{j=0}^{n}B_j^n(x)+\frac{\|f\|_I}{2n\delta^2}=\frac{\varepsilon}{2}+\frac{\|f\|_I}{2n\delta^2}.\end{aligned}$$

Die rechte Seite ist nun unabhängig von x und $< \varepsilon$ sobald $n > \varepsilon^{-1}\;\delta^{-2}\;\|f\|_I$. □

Der Beweis zeigt uns, daß die Approximationsgeschwindigkeit (also der Grad, der für eine bestimmte Approximationsgüte notwendig ist) entscheidend davon abhängt, wie gleichmäßig stetig die Funktion f, denn der Wert $\delta = \delta(\varepsilon)$ kann beliebig schnell fallen. Dies ist ein weiterer Beleg dafür, daß Stetigkeit allein oftmals nicht genug ist, ein generelles Prinzip bei Anwendungen.

7.1 Darstellungen polynomialer Kurven und Auswertungsverfahren

Die wichtigste Operation beim praktischen Rechnen mit Funktionen ist sicherlich die *Auswertung* dieser Funktion an einer Stelle $x \in \mathbb{R}$. Dieses Auswertungsverfahren hängt sicherlich von

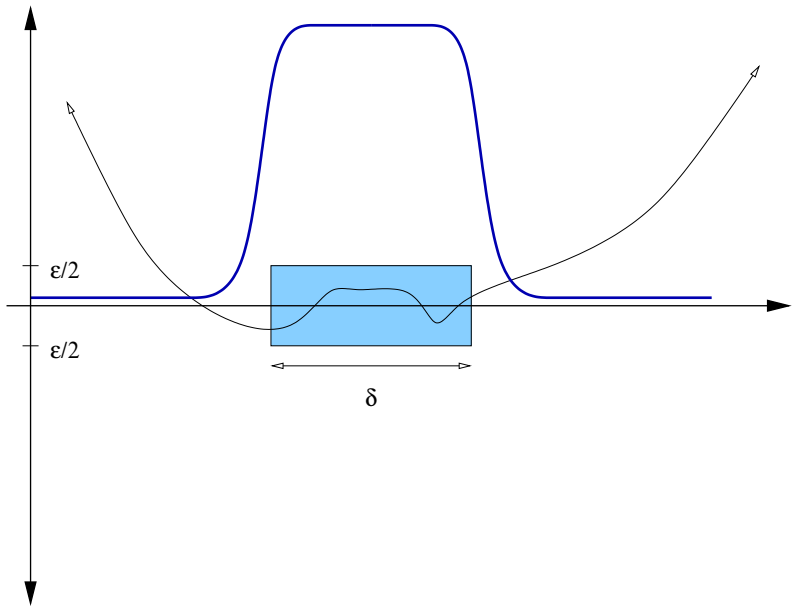


Abbildung 7.1: Idee des Beweises von Satz 7.3: Auf dem farbig unterlegten δ–Intervall um den Punkt x weichen die Funktionen um maximal ε voneinander ab, und diese Abweichung ist wegen der gleichmäßigen Stetigkeit unabhängig von der Lage des δ–"Fensters", auf dem Rest des Intervalls hat die (blaue) "Maskierungsfunktion" hingegen so einen kleinen Wert, daß der Fehler dort machen kann, was er will.

der verwendeten Darstellung ab. Im CAGD[50] verwendet man nicht nur polynomiale *Funktionen* sondern polynomiale *Kurven*.

Definition 7.4 *Eine Abbildung*

$$\boldsymbol{p} = \begin{bmatrix} p_1 \\ \vdots \\ p_d \end{bmatrix} \quad : \quad \mathbb{R} \to \mathbb{R}^d$$

heißt polynomiale Kurve *vom (Höchst–) Grad n, falls $p_j \in \Pi_n$, $j = 1, \ldots, d$. In diesem Fall schreiben wir auch $\boldsymbol{p} \in \Pi_n^d$.*

Bemerkung 7.5 *Jede polynomiale Kurve $\boldsymbol{p} \in \Pi_n^d$ läßt sich schreiben als*

$$\boldsymbol{p}(x) = \begin{bmatrix} a_{10} + a_{11}x + \cdots + a_{1n}x^n \\ \vdots \\ a_{d0} + a_{d1}x + \cdots + a_{dn}x^n \end{bmatrix} = \underbrace{\begin{bmatrix} a_{10} \\ \vdots \\ a_{d0} \end{bmatrix}}_{=\boldsymbol{a}_0} + \underbrace{\begin{bmatrix} a_{11} \\ \vdots \\ a_{d1} \end{bmatrix}}_{=\boldsymbol{a}_1} x + \cdots + \underbrace{\begin{bmatrix} a_{1n} \\ \vdots \\ a_{dn} \end{bmatrix}}_{=\boldsymbol{a}_n} x^n$$

[50] **C**omputer **A**ided **G**eometric **D**esign.

$$= \sum_{j=0}^{n} \boldsymbol{a}_j \, x^j,$$

wobei $\boldsymbol{a}_j \in \mathbb{R}^d$, $j = 1, \ldots, n$.

Proposition 7.6 *(Horner–Schema)*
Ist

$$\boldsymbol{p}(x) = \sum_{j=0}^{n} \boldsymbol{a}_j x^j \qquad \in \Pi_n^d,$$

und setzt man

$$\boldsymbol{a}^0(x) = \boldsymbol{a}_n \tag{7.4}$$
$$\boldsymbol{a}^j(x) = x \, \boldsymbol{a}^{j-1}(x) + \boldsymbol{a}_{n-j}, \qquad j = 1, \ldots, n, \tag{7.5}$$

dann ist $\boldsymbol{a}^n(x) = \boldsymbol{p}(x)$.

Beweis: Schreibe $\boldsymbol{p}(x)$ als

$$\boldsymbol{p}(x) = \boldsymbol{a}_0 + x \, (\boldsymbol{a}_1 + x \, (\cdots x \, (\boldsymbol{a}_{n-1} + x \, \boldsymbol{a}_n) \cdots))$$

und werte von innen nach außen aus. □

Das Hornerschema findet sich in `Horner.m`.

Mit genau derselben Faktorisierungsidee funktioniert das Horner–Schema auch für eine größere Klasse von polynomialen Funktionen.

Korollar 7.7 *Es seien*

$$\ell_j(x) = \alpha_j x - \beta_j \ \in \Pi_1, \quad \alpha_j \neq 0, \qquad j \in \mathbb{N},$$

und[51]

$$p_j(x) = \prod_{k=1}^{j} \ell_k(x), \qquad j \in \mathbb{N}_0,$$

sowie

$$\boldsymbol{p}(x) = \sum_{j=0}^{n} \boldsymbol{a}_j \, p_j(x) \qquad \in \Pi_n^d.$$

Dann liefert, für jedes $x \in \mathbb{R}$ *die Rekursion*

$$\boldsymbol{a}^0(x) = \boldsymbol{a}_n \tag{7.6}$$
$$\boldsymbol{a}^j(x) = \ell_{n-j+1}(x) \, \boldsymbol{a}^{j-1}(x) + \boldsymbol{a}_{n-j}, \qquad j = 1, \ldots, n, \tag{7.7}$$

als Ergebnis $\boldsymbol{a}^n(x) = \boldsymbol{p}(x)$.

[51] Dabei ist $p_0 = 1$. Das entspricht der Konvention, daß leere Produkte den Wert 1 haben.

```
%%  Horner.m (Numerik 1)
%%  ----------------------------------------------------
%%  Hornerschema
%%  Eingabe:
%%    a    Koeffizientenmatrix (vektorwertig!)
%%    x    Stelle

function y = Horner( a,x )
  n = length(a);
  b = a(:,n);

  for j = n-1:-1:1
    b = x * b + a(:,j);
  end

  y = b;
%endfunction
```

Programm 7.1 `Horner.m`: Das Hornerschema für vektorwertige Polynome.

Als nächstes sehen wir uns die numerische Stabilität des Hornerschemas an, und zwar für die allgemeine Form.

Satz 7.8 *Sei $x \in \mathbb{F}$. Wenn es eine Konstante $\eta > 0$ gibt, so daß*

$$\alpha_j \otimes x \ominus \beta_j = \ell_j(x)\ (1+\delta_j)\,, \qquad |\delta_j| \le \eta\ \hat{u}, \quad j = 1, \ldots, n, \tag{7.8}$$

dann erfüllt der mit dem Hornerschema berechnete Wert

$$\hat{\boldsymbol{p}}(x) = \sum_{j=0}^{n} \hat{\boldsymbol{a}}_j\ p_j(x)$$

die Ungleichung

$$|\hat{\boldsymbol{a}}_j - \boldsymbol{a}_j| \le \big((1+(\eta+2)j)\ \hat{u} + O\big(\hat{u}^2\big)\big)\,|\boldsymbol{a}_j|\,. \tag{7.9}$$

Bevor wir den Satz beweisen, erst einmal ein paar wichtige Bemerkungen:

1. Gleichung (7.8) ist eine wichtige Bedingung, die im wesentlichen verlangt, daß nicht gerade $\alpha_j x \sim \beta$[52] ist. Allerdings: So dramatisch ist es auch wieder nicht, denn beispielsweise in den Fällen

   - $\alpha = 1$,

[52] Stichwort: Auslöschung!

- $\beta = 0$,
- $\alpha\ \beta\ x < 0$

ist (7.8) automatisch erfüllt, zum Teil sogar mit $\eta = 1$.

2. Im Falle des "klassischen" Hornerschemas ist sogar $\eta = 0$ (die "Rechnung" ist exakt).

3. Die Fehlerabschätzung (7.9) sagt uns, daß die Koeffizienten eines Polynoms unterschiedlich zum Rundungsfehler beitragen und zwar umso mehr, je höher der Grad des zugehörigen Monoms ist. Generell heißt das auch, daß wachsender Grad die Auswertung von Polynomen instabiler machen wird.

4. Es gibt auch Beschreibungen des Hornerschemas für *multivariate* Polynome [31, 32], also Polynome in sagen wir s Variablen, bei dem man die rekursive Natur so ausnutzen kann, daß der Rückwärtsfehler linear im Grad n und nicht in der Anzahl der Koeffizienten[53] $\binom{n+s}{s} \sim n^s$ ist – für große Werte von s ist das ein ganz gewaltiger Unterschied.

Beweis: Aus den Rekursionsformeln (7.6) und (7.7) sehen wir sofort, daß

$$\boldsymbol{a}^j(x) = \sum_{k=0}^{j} \boldsymbol{a}_{n-j+k} \prod_{r=1}^{k} \ell_{n-j+r}(x),$$

und behaupten, daß

$$\hat{\boldsymbol{a}}^j(x) = \sum_{k=0}^{j} \hat{\boldsymbol{a}}^j_{n-j+k} \prod_{r=1}^{k} \ell_{n-j+r}(x), \tag{7.10}$$

wobei

$$\hat{\boldsymbol{a}}^j_{n-j+k} = (1+\gamma_k)\,\boldsymbol{a}_{n-j+k}, \qquad |\gamma_k| \le (1+(\eta+2)k)\,\hat{u} + O\left(\hat{u}^2\right), \quad k = 0, \ldots, j. \tag{7.11}$$

Für $j = 0$ sind (7.10) und (7.11) wegen (7.6) trivialerweise erfüllt. Ansonsten haben wir nach (7.7), daß für $j = 0, \ldots, n-1$

$$\begin{aligned}
\hat{\boldsymbol{a}}^{j+1}(x) &= \hat{\ell}_{n-j}(x) \otimes \hat{\boldsymbol{a}}^j(x) \oplus \boldsymbol{a}_{n-(j+1)} \\
&= \left(\ell_{n-j}(x)\ (1+\delta)\ \hat{\boldsymbol{a}}^j(x)\,(1+\varepsilon) + \boldsymbol{a}_{n-j-1}\right)(1+\theta) \\
&= \ell_{n-j}(x)\ \left((1+\delta)(1+\varepsilon)(1+\theta)\ \hat{\boldsymbol{a}}^j(x)\right) + (1+\theta)\,\boldsymbol{a}_{n-j-1},
\end{aligned}$$

wobei $|\delta| \le \eta\hat{u}$ und $|\varepsilon|, |\theta| < \hat{u}$; also ist, für ein γ mit $|\gamma| \le (\eta+2)\hat{u}$,

$$\begin{aligned}
\hat{\boldsymbol{a}}^{j+1}(x) &= \left(1+\gamma+O\left(\hat{u}^2\right)\right)\ \ell_{n-j}(x)\hat{\boldsymbol{a}}^j(x) + (1+\theta)\,\boldsymbol{a}_{n-j} \\
&= \sum_{k=0}^{j} \underbrace{\left(1+\gamma+O\left(\hat{u}^2\right)\right)\hat{\boldsymbol{a}}^j_{n-j+k}}_{=:\hat{\boldsymbol{a}}^{j+1}_{n-j+k} = \hat{\boldsymbol{a}}^{j+1}_{n-(j+1)+(k+1)}} \prod_{r=0}^{k} \ell_{n-(j+1)+(r+1)}(x) + \underbrace{(1+\theta)\,\boldsymbol{a}_{n-j-1}}_{=:\hat{\boldsymbol{a}}^{j+1}_{n-(j+1)}} \\
&= \sum_{k=0}^{j+1} \hat{\boldsymbol{a}}^{j+1}_{n-(j+1)+k} \prod_{r=1}^{k} \ell_{n-(j+1)+r}(x),
\end{aligned}$$

[53] Und damit Rechenoperationen!

wobei, für $k = 1, \ldots, j+1$

$$\begin{aligned}\hat{\boldsymbol{a}}_{n-(j+1)+k}^{j+1} &= \left(1+\gamma+O\left(\hat{u}^2\right)\right)\left(1+\gamma_{k-1}\right)\boldsymbol{a}_{n-(j+1)+k} \\ &= \left(1+\gamma_{k-1}+\gamma+O\left(\hat{u}^2\right)\right)\boldsymbol{a}_{n-(j+1)+k} =: \left(1+\gamma_k\right)\boldsymbol{a}_{n-(j+1)+k},\end{aligned}$$

mit

$$|\gamma_k| \leq |\gamma_{k-1}| + |\gamma| + O\left(\hat{u}^2\right) \leq 1 + (\eta+2)(k-1) + (\eta+2) + O\left(\hat{u}^2\right) = 1 + (\eta+2)k + O\left(\hat{u}^2\right).$$

Für $k = 0$ ist (7.7) offensichtlich. □

Im CAGD wurden das Hornerschema und damit die monomiale Darstellung von polynomialen Kurven aber inzwischen von etwas anderem verdrängt. Der Grund ist, daß die Koeffizienten $\boldsymbol{a}_j = \frac{1}{j!}\boldsymbol{p}^{(j)}(0)$, $j = 0, \ldots, n$, kaum *geometrische* Information über die resultierende Kurve geben und daß das Design mit diesen Koeffizienten praktisch unmöglich ist.

Definition 7.9 *Es sei $I = [t_0, t_1] \subset \mathbb{R}$, $-\infty < t_0 < t_1 < \infty$, ein nichtdegeneriertes kompaktes Intervall*[54] *und $x \in \mathbb{R}$. Die* baryzentrischen Koordinaten

$$u\left(x \mid I\right) = \left(u_0(x \mid I), u_1(x \mid I)\right)$$

von x bezüglich I sind definiert als

$$x = u_0\left(x \mid I\right)\ t_0 + u_1\left(x \mid I\right)\ t_1, \qquad u_0\left(x \mid I\right) + u_1\left(x \mid I\right) = 1. \tag{7.12}$$

Bemerkung 7.10 *(Baryzentrische Koordinaten)*

1. *In (7.12) wird x als* affine Kombination *von t_0 und t_1 dargestellt. Eine affine Kombination, bei der beide Koeffizienten ≥ 0 sind, bezeichnet man als* Konvexkombination.

2. *Die baryzentrischen Koordinaten bestimmen sich als*

$$u_0\left(x \mid I\right) = \frac{t_1 - x}{t_1 - t_0} \qquad \textit{und} \qquad u_1\left(x \mid I\right) = \frac{x - t_0}{t_1 - t_0}. \tag{7.13}$$

3. *Aus (7.13) folgt sofort, daß*

$$u_0\left(t_j \mid I\right) = \delta_{0j} \qquad \textit{und} \qquad u_1\left(t_j \mid I\right) = \delta_{1j}, \qquad j = 0, 1. \tag{7.14}$$

4. *Die baryzentrischen Koordinaten*[55] *beschreiben das Verhältnis, in dem der Punkt x das Intervall I teilt, siehe Abb. 7.2.*

5. *Es gilt*

$$u_0\left(x \mid I\right), u_1\left(x \mid I\right) \geq 0 \qquad \Longleftrightarrow \qquad x \in I.$$

[54] Also das, was sich der "Mann von der Straße" so unter einem Intervall vorstellt.

[55] Der Begriff bedeutet ja nichts anderes als "Schwerpunktskoordinaten".

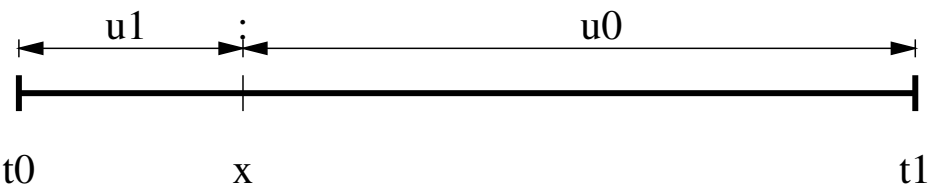


Abbildung 7.2: Baryzentrische Koordinaten als (normiertes) Teilungsverhältnis.

6. *Ist $I = [0, 1]$, dann ist $u_0(x \mid I) = (1-x)$ und $u_1(x \mid I) = x$. Baryzentrische Koordinaten transformieren also das Intervall I in $[0, 1]$.*

Mit Hilfe der baryzentrischen Koordinaten können wir ein anderes Auswertungsverfahren für Polynome angeben.

Satz 7.11 *Es seien $\boldsymbol{b}_0, \ldots, \boldsymbol{b}_n \in \mathbb{R}^d$. Für $x \in I$ liefert der* Algorithmus von de Casteljau[56],

$$\boldsymbol{b}_k^0(x) = \boldsymbol{b}_k, \qquad k = 0, \ldots, n, \tag{7.15}$$

$$\boldsymbol{b}_k^{j+1}(x) = u_0(x \mid I)\ \boldsymbol{b}_k^j(x) + u_1(x \mid I)\ \boldsymbol{b}_{k+1}^j(x), \qquad k = 0, \ldots, n-j-1, \tag{7.16}$$

als Ergebnis den Wert

$$\boldsymbol{b}_0^n(x) = \sum_{j=0}^{n} \boldsymbol{b}_j \underbrace{\binom{n}{j} u_0^{n-j}(x \mid I)\ u_1^j(x \mid I)}_{=:B_j^n(x \mid I)} \tag{7.17}$$

der Bézierkurve[57]

$$B_n\boldsymbol{b}(x) := B_n(\boldsymbol{b}_0, \ldots, \boldsymbol{b}_n)(x) = \sum_{j=0}^{n} \boldsymbol{b}_j\ B_j^n(x \mid I)$$

an der Stelle x.

Definition 7.12 *Die Koeffizienten $\boldsymbol{b} = (\boldsymbol{b}_0, \ldots, \boldsymbol{b}_n)$ bezeichnet man als* Kontrollpunkte*, die von ihnen gebildete stückweise lineare Funktion als* Kontrollpolygon*.*

Die schematische Arbeitsweise des Algorithmus von de Casteljau ist in Abb. 7.3 dargestellt.

Die Polynome $B_j^n(x \mid I)$ sind im Falle $I = [0, 1]$ "alte Bekannte", nämlich genau die Polynome, mit deren Hilfe wir die Bernsteinpolynome im Beweis von Satz 7.3 gebildet haben. Tatsächlich ist ja $B_n f$ auch nichts anderes als eine Bézierkurve, nur eben mit $\boldsymbol{b}_j = f(j/n)$, $j = 0, \ldots, n$. Aus disem Grund bezeichnet man die Polynome $B_j^n(x \mid I)$ auch als *Bernstein–Bézier–Basispolynome*.

Übung 7.1 Zeigen Sie, daß jedes stetige Vektorfeld $\boldsymbol{f} : I \to \mathbb{R}^d$ auf dem kompakten Intervall $I \subset \mathbb{R}$ gleichmäßig durch polynomiale Vektorfelder approximiert werden kann.

[56] Paul Faget de Casteljau, Mathematiker, in den 60ern bei Citröen in der CAD–Entwicklung tätig.
[57] Pierre Bézier, † 29.11.1999, Ingenieur, in den 60ern bei Renault in der CAD–Entwicklung tätig.

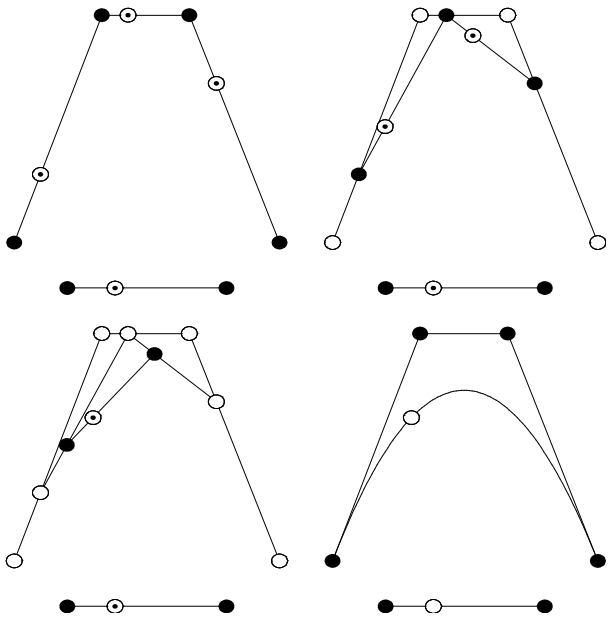


Abbildung 7.3: Der Algorithmus von de Casteljau: Die Teilungsverhältnisse von I bezüglich x werden auf die Streckenzüge übertragen, die die Koeffizienten verbinden.

Lemma 7.13 *Die Bernstein–Bézier–Basispolynome haben die folgenden Eigenschaften:*

1. *(Rekursionsformel)*

$$B_j^n(x \mid I) = u_0(x \mid I)\ B_j^{n-1}(x \mid I) + u_1(x \mid I)\ B_{j-1}^{n-1}(x \mid I). \tag{7.18}$$

2. *(Teilung der Eins)*

$$\sum_{j=0}^{n} B_j^n(x \mid I) = 1. \tag{7.19}$$

3. *(Nichtnegativität) Für $x \in I$ ist $B_j^n(x \mid I) \geq 0$.*

Beweis: Die erste Identität, (7.18), folgt aus[58]

$$\begin{aligned} B_j^n(x \mid I) &= \binom{n}{j} u_0^{n-j} u_1^j = \left(\binom{n-1}{j} + \binom{n-1}{j-1} \right) u_0^{n-j} u_1^j \\ &= u_0 \binom{n-1}{j} u_0^{n-1-j} u_1^j + u_1 \binom{n-1}{j-1} u_0^{n-j} u_1^{j-1} \\ &= u_0(x \mid I)\ B_j^{n-1}(x \mid I) + u_1(x \mid I)\ B_{j-1}^{n-1}(x \mid I). \end{aligned}$$

[58] Hier und im Folgenden verwenden wir die Kurzschreibweise $u_j = u_j(x \mid I)$, $x \in I$.

```
%%  DeCasteljau.m (Numerik I)
%%  -----------------------------------------------------
%%  DeCasteljau fuer [0,1]
%%  Eingabe:
%%    b    Kontrollpunkte (Matrix!)
%%    x    Punkt in [0,1]

function y = DeCasteljau( b,x )
  n = length(b);

  for j = 1:n-1
    for k = 1:n-j
      b( :,k ) = (1-x) * b( :,k ) + x * b( :,k+1 );
    end
  end

  y = b( :,1 );
%endfunction
```

Programm 7.2 `DeCasteljau.m`: Der Algorithmus von de Casteljau mit Referenzintervall $I = [0,1]$.

Für (7.19) verwenden wir (7.18) und erhalten mit der Konvention $B^n_{-1} = B^n_{n+1} = 0$, daß

$$\sum_{j=0}^{n} B^n_j\,(x \mid I) = \sum_{j=0}^{n-1} u_0 B^{n-1}_j + \sum_{j=1}^{n} u_1 B^{n-1}_{j-1} = \underbrace{(u_0+u_1)}_{=1} \underbrace{\sum_{j=0}^{n-1} B^{n-1}_j}_{=1} = 1$$

mittels Induktion über n. Die Nichtnegativität folgt direkt aus (7.18). □

Übung 7.2 Beweisen Sie die kombinatorische Identität

$$\binom{n}{j} = \binom{n-1}{j} + \binom{n-1}{j-1}.$$

Beweis von Satz 7.11: Wir zeigen durch Induktion über j, daß

$$\boldsymbol{b}^j_k(x) = \sum_{r=0}^{j} \boldsymbol{b}_{k+r}\; B^j_r\,(x \mid I)\,, \qquad k = 0,\ldots,n-j,\; x \in \mathbb{R}, \tag{7.20}$$

was für $j = n$ die Formel (7.17) liefert und für $j = 0$ gerade (7.15), also per Definitionem richtig ist. Für $j \geq 0$ und $k = 0,\ldots,n-j-1$ verwenden wir schließlich die Rekursionsformel (7.16), um mit Hilfe der Induktionssanahme

$$\begin{aligned}
\boldsymbol{b}^{j+1}_k(x) &= u_0\,\boldsymbol{b}^j_k(x) + u_1\,\boldsymbol{b}^j_{k+1} = u_0 \sum_{r=0}^{j} \boldsymbol{b}_{k+r} B^j_r + u_1 \sum_{r=0}^{j} \boldsymbol{b}_{k+r+1} B^j_r \\
&= u_0 \sum_{r=0}^{j} \boldsymbol{b}_{k+r} B^j_r + u_1 \sum_{r=1}^{j+1} \boldsymbol{b}_{k+r} B^j_{r-1} = \sum_{r=0}^{j+1} \boldsymbol{b}_{k+r} \left(u_0 B^j_r + u_1 B^j_{r-1}\right) \\
&= \sum_{r=0}^{j+1} \boldsymbol{b}_{k+r} B^{j+1}_r\,(x \mid I)
\end{aligned}$$

zu erhalten, was die Induktion fortsetzt. □

Bemerkung 7.14 *(Eigenschaften der Bézierkurven)*

1. *("Convex Hull Property") Da für $x \in I$*

$$0 \leq B^n_j\,(x \mid I) \qquad \textit{und} \qquad \sum_{j=0}^{n} B^n_j\,(x \mid I) = 1,$$

ist jeder Punkt $B_n\boldsymbol{b}$ eine Konvexkombination *von $\boldsymbol{b}_0,\ldots,\boldsymbol{b}_n$ und damit liegt die Kurve*[59] *$B_n\boldsymbol{b}(I)$ in der konvexen Hülle von $\boldsymbol{b}_0,\ldots,\boldsymbol{b}_n$. Dies ist beispielsweise für die Schnittbestimmung von Kurven wichtig: Man prüft zuerst einmal, ob sich überhaupt die konvexen Hüllen (also Polygone*[60]*) schneiden, bevor man ein kompliziertes Verfahren zur Schnittberechnung von Kurven anwirft.*

[59] Genauer: ihr Graph.

[60] Zur Bestimmung des Schnitts von Polygonen sind eine Vielzahl von sehr effizienten Algorithmen, Stichwort "Clipping", verfügbar.

2. *("Endpoint Interpolation") Ist* $u_0\,(x \mid I) = 1$*, dann ist* $\boldsymbol{b}_0^j(x) = \boldsymbol{b}_0$*,* $j = 0, \ldots, n$*, also auch* $B_n\boldsymbol{b}(x) = \boldsymbol{b}_0$*; entsprechend natürlich auch* $B_n\boldsymbol{b}(x) = \boldsymbol{b}_n$*, wenn* $u_1\,(x \mid I) = 1$*. Die Bézierkurven interpolieren also an den Endpunkten des Kontrollpolygons.*

3. *("Linear Precision") Liegen alle Kontrollpunkte* $\boldsymbol{b}_j$*,* $j = 0, \ldots, n$*, auf einer Geraden, so ist* $B_n\boldsymbol{b}(I)$ *ein Streckenzug auf dieser Geraden: Für alle* $x \in \mathbb{R}$ *mit* $\boldsymbol{b}_k^j(x)$ *und* $\boldsymbol{b}_{k+1}^j(x)$ *liegt auch der Zwischenpunkt* $\boldsymbol{b}_k^{j+1}(x)$ *auf der Geraden. Allerdings ist im Normalfall die Kurve* $B_n\boldsymbol{b}(x)$ *anders parametrisiert als der Streckenzug. Zum Beispiel durchläuft für*

$$\boldsymbol{b}_0 = \boldsymbol{b}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{b}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

die Kurve $B_2\boldsymbol{b}$ *den Streckenzug "hin und zurück", siehe Abb. 7.4.*

4. *Die Form des Kontrollpolygons beschreibt "ungefähr" die Form der resultierenden Kurve.*

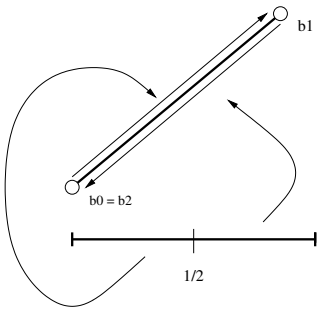


Abbildung 7.4: Eine Bézierkurve, deren Graph ein Streckenzug ist, der aber zweimal durchlaufen wird.

Auch der Rückwärtsfehler des Algorithmus von de Casteljau hält sich in vernünftigem Rahmen.

Satz 7.15 *Kann man die baryzentrischen Koordinaten von* x *mit der Genauigkeit*

$$\hat{u}_j = u_j\,(x \mid I)\,(1 + \varepsilon_j)\,, \qquad |\varepsilon_j| \le \eta\,\hat{u}, \quad j = 1, 2,$$

bestimmen, dann ist der berechnete Wert

$$\widehat{B}_n\boldsymbol{b}(x) = \sum_{j=0}^{n} \hat{\boldsymbol{b}}_j\; B_j^n\,(x \mid I)\,,$$

wobei

$$\hat{\boldsymbol{b}}_j = \boldsymbol{b}_j\,(1 + \delta_j)\,, \qquad |\delta_j| \le n\,(\eta + 2)\,\hat{u} + O\left(\hat{u}^2\right). \tag{7.21}$$

Beweis: Aus der Rekursionsformel (7.16) folgt, daß für $j < n$ und $k = 0, \ldots, n-j-1$

$$\begin{aligned}
\hat{\boldsymbol{b}}_k^{j+1}(x) &= \hat{u}_0 \otimes \boldsymbol{b}_k^j(x) \oplus \hat{u}_1 \otimes \boldsymbol{b}_{k+1}^j(x) \\
&= \left(\hat{u}_0\ \hat{\boldsymbol{b}}_k^j(x)(1+\theta_j) + \hat{u}_1\ \hat{\boldsymbol{b}}_{k+1}^j(x)\left(1+\theta_j'\right)\right)(1+\gamma_j) \\
&= \hat{u}_0 \hat{\boldsymbol{b}}_k^j(x)\,(1+\theta_j)\,(1+\gamma_j) + \hat{u}_0 \hat{\boldsymbol{b}}_k^j(x)\left(1+\theta_j'\right)(1+\gamma_j)\,.
\end{aligned}$$

Aus dieser Identität zeigt man dann per Induktion[61], daß[62]

$$\hat{\boldsymbol{b}}_k^j(x) = \sum_{r=0}^{j} \hat{\boldsymbol{b}}_r^j(x)\ \binom{j}{r} \hat{u}_0^{j-r}\ \hat{u}_1^r = \sum_{r=0}^{j} \hat{\boldsymbol{b}}_r^j(x)\ B_j^n\left(\hat{u}_1\right), \tag{7.22}$$

wobei

$$\hat{\boldsymbol{b}}_r^j(x) = \boldsymbol{b}_r^j(x)\ \left(1+\delta_{j,r}\right), \qquad |\delta_{j,r}| \le 2j\hat{u} + O\left(\hat{u}^2\right). \tag{7.23}$$

Da außerdem

$$\begin{aligned}
B_j^n\left(\hat{u}_1\right) &= \binom{n}{j} \left((1+\varepsilon_0)\ u_0\left(x \mid I\right)\right)^{n-j} \left((1+\varepsilon_1)\ u_1\left(x \mid I\right)\right)^j \\
&= (1+\varepsilon_0)^{n-j}\,(1+\varepsilon_1)^j\ B_j^n\left(x \mid I\right) = (1+\varepsilon)\ B_j^n\left(x \mid I\right),
\end{aligned}$$

wobei $|\varepsilon| \le n\eta\ \hat{u} + O\left(\hat{u}^2\right)$, ergibt sich (7.21). □

Neben den ansprechenden geometrischen Eigenschaften haben Bézierkurven auch gute numerische Eigenschaften.

Definition 7.16 *Es sei $\mathscr{P}_n = \{p_0, \ldots, p_n\}$ eine Basis von Π_n und $I \subset \mathbb{R}$. Die* globale Konditionszahl $\kappa_I\left(\mathscr{P}_n\right)$ *ist definiert als*

$$\kappa_I\left(\mathscr{P}_n\right) = \max_{x \in I} \max_{|c_j| \le 1} \sum_{j=0}^{n} |c_j\ p_j(x)| = \max_{x \in I} \sum_{j=0}^{n} |p_j(x)|\,. \tag{7.24}$$

Satz 7.17 *(Konditionszahlen und Fehler)*

1. *Sei $x \in I$ und erfülle der mit einem numerischen Verfahren berechnete Wert $\hat{\mathbf{p}}(x)$ für ein $\eta > 0$ die Bedingung*[63]

$$\hat{\boldsymbol{p}}(x) = \sum_{j=0}^{n} \hat{\boldsymbol{a}}_j\ p_j(x), \qquad \hat{\boldsymbol{a}}_j = \boldsymbol{a}_j\left(1+\delta_j\right), \quad |\delta_j| \le \eta\ \hat{u}, \tag{7.25}$$

[61] Wer es nachvollziehen will: Die Sache wird einfacher unter der (abwegigen) Annahme, daß $\hat{u}_0 + \hat{u}_1 = 1$, ansonsten gibt's aber auch nur einen $O\left(\hat{u}^2\right)$–Term.

[62] Ich denke mal, daß mir niemand böse ist, wenn wir die Details hier weglassen, die Sache ist wirklich "straightforward".

[63] Nichts anderes als ein Rückwärtsfehler.

dann ist[64]

$$|\boldsymbol{p}(x) - \hat{\boldsymbol{p}}(x)| \leq \kappa_I\,(p_0, \ldots, p_j)\ \eta\ \hat{u}\ |\boldsymbol{a}|\,, \tag{7.26}$$

wobei

$$|\boldsymbol{a}| = \left(\max_{j=0,\ldots,n} |a_{jk}|\ :\ k = 1, \ldots, d\right), \qquad \boldsymbol{a}_j = (a_{jk} : k = 1, \ldots, d)\,.$$

2. *Für* $I = [0, 1]$ *ist*

$$\kappa_I\,(1, x, \ldots, x^n) = n + 1 \tag{7.27}$$

und

$$\kappa_I\,(B_0^n(x \mid I), \ldots, B_n^n(x \mid I)) = 1. \tag{7.28}$$

Beweis: Es ist

$$|\boldsymbol{p}(x) - \hat{\boldsymbol{p}}(x)| = \left|\sum_{j=0}^{n} \delta_j \boldsymbol{a}_j\ p_j(x)\right| \leq \eta\ \boldsymbol{a}\ \sum_{j=0}^{n} |p_j(x)| \leq \kappa_I\,(p_0, \ldots, p_j)\ \eta\ \hat{u}\ \boldsymbol{a}.$$

Außerdem ist das Polynom $p(x) = 1 + x + \cdots + x^n$ monoton steigend auf $[0, 1]$, also ist $\kappa_I\,(1, x, \ldots, x^n) = p(1) = n + 1$ und (7.28) ist nichts anderes als (7.19). □

Bemerkung 7.18 *Man kann sogar zeigen, daß die Bernstein–Bézier–Basis* optimale *Stabilität besitzt, das heißt, daß sie für numerische Rechnungen die höchste Genauigkeit liefert [15, 14]. Allerdings mit Einschränkungen: Dies gilt nur, wenn man die Polynome auf dem baryzentrischen Einheitsintervall betrachtet*[65] *und man muß von Anfang an in dieser Basis rechnen. Eine Konvertierung von Polynomen aus einer anderen Basis, beispielsweise der Monombasis, in die Bernstein–Bézier–Basis verursacht größere Fehler, als man durch die Wahl der besseren Basis vermeiden könnte. Es gibt eben nichts umsonst.*

7.2 Interpolation – der Aitken–Neville-Algorithmus

In vielen Anwendungen spielt das folgende *Interpolationsproblem* eine wichtige Rolle:

> *Gegeben seien Punkte* $x_0, \ldots, x_n \in I \subset \mathbb{R}$, I *ein kompaktes Intervall. Zu vorgegebenen Werten* $\boldsymbol{f}_0, \ldots, \boldsymbol{f}_n \in \mathbb{R}^d$ *finde man eine stetige Funktion* $\boldsymbol{f} : I \to \mathbb{R}^d$, *so daß*
>
> $$\boldsymbol{f}\,(x_j) = \boldsymbol{f}_j, \qquad j = 0, \ldots, n.$$

[64] Dies ist eine Abschätzung für den *absoluten* Fehler der Auwertung; der relative Fehler in Komponente j kann beliebig übel werden, wenn $\boldsymbol{p}_j(x) \sim 0$.

[65] Also immer *im* Referenzintervall bleibt – sobald man dieses verläßt, verschlechtert sich die Situation dramatisch.

Derartige mathematische Aufgaben entstehen oder enstanden zum Beispiel bei der Erstellung ballistischer Tabellen, bzw. bei der Approximation heuristischer (d.h. gemessener) Tabellen[66]

Natürlich hat das obige Interpolationsproblem unendlich viele Lösungen. Die "prominenteste" und "klassischste" sind sicherlich Polynome.

Satz 7.19 *Zu vorgegebenen* Knoten $x_0 < x_1 < \cdots < x_n$ *und* Daten $\boldsymbol{f}_0, \ldots, \boldsymbol{f}_n$ *gibt es genau ein Polynom* $\boldsymbol{p} \in \Pi_n$*, so daß*

$$\boldsymbol{p}\,(x_j) = \boldsymbol{f}_j, \qquad j = 0, \ldots, n. \tag{7.29}$$

Der "Standardbeweis" für die obige Aussage verwendet die *Lagrange–Darstellung*

$$\boldsymbol{p}(x) = \sum_{j=0}^{n} \boldsymbol{f}_j \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k},$$

die aber numerisch *hochgrading* instabil ist. Wir wollen hier einen anderen, algorithmischen Weg gehen. Dazu schreiben wir

$$I_k^j := [x_k, x_{k+j}]\,, \qquad k = 0, \ldots, n-j,\ j = 1, \ldots, n.$$

Satz 7.20 *Für vorgegebene* $x_0 < x_1 < \cdots < x_n \in \mathbb{R}$ *und* $\boldsymbol{f}_0, \ldots, \boldsymbol{f}_n \in \mathbb{R}^d$ *liefert der Aitken–Neville–Algorithmus*[67]

$$\begin{aligned} \boldsymbol{f}_k^0(x) &= \boldsymbol{f}_k, \qquad k = 0, \ldots, n & (7.30)\\ \boldsymbol{f}_k^{j+1}(x) &= u_0\left(x \mid I_k^{j+1}\right)\ \boldsymbol{f}_k^j(x) + u_1\left(x \mid I_k^{j+1}\right)\ \boldsymbol{f}_{k+1}^j(x), \quad k = 0, \ldots, n-j-1, & (7.31) \end{aligned}$$

als Endergebnis ein Polynom $\boldsymbol{p}(x) = \boldsymbol{f}_0^n(x) \in \Pi_n^d$ *mit der Interpolationseigenschaft*

$$\boldsymbol{p}\,(x_j) = \boldsymbol{f}_j, \qquad j = 0, \ldots, n. \tag{7.32}$$

Der Algorithmus von Aitken–Neville sieht eigentlich ganz genauso aus wie der Algorithmus von de Casteljau – mit einem kleinen, aber feinen und signifikanten Unterschied: In jedem Schritt von Aitken–Neville wird ein anderes Referenzintervall verwendet! Trotzdem ist Aitken–Neville, wie auch der Algorithmus von de Casteljau, ein sogenanntes "*Pyramidenschema*". Der Name stammt daher, daß das Auswertungsschema, grafisch dargestellt, eine auf dem Kopf stehende Pyramide ergibt, wenn man mit den Ausgangsparametern beginnt und dann Zeile für

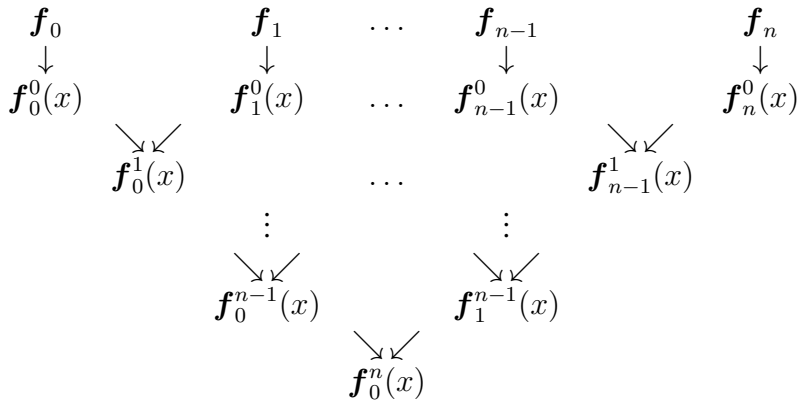
[66] Genau diese Problematik führte I. J. Schoenberg zur "Erfindung" der Splines während seiner Tätigkeit in den "Ballistic Research Laboratories", Aberdeen, Maryland, während des zweiten Weltkriegs.

[67] Zuerst von A. G. Aitken (1932) eingeführt und später auch von E. H. Neville (1934) untersucht, siehe [1, 27]. Insbesondere wird in [27] *explizit* auf [1] verwiesen.

```
%%  AitkenNeville.m
%%  ------------------------------------------------------
%%  Aitken-Neville-Interpolation
%%  Eingabe:
%%    f    Funktionswerte
%%    X    Interpolationsknoten
%%    x    Asuwertungsstelle

function y = AitkenNeville( f,X,x )
  n = length( f );

  for j=1:n
    for k=1:n-j
      t = ( X( k+j ) - x ) / ( X( k+j ) - X( k ) );
      f( :,k ) = t * f( :,k ) + ( 1-t ) * f( :,k+1 );
    end
  end

  y = f( :,1 );
%endfunction
```

Programm 7.3 `AitkenNeville.m`: Der Algorithmus von Aitken–Neville. Die Matrix `f` enthält die zu interpolierenden Werte, der Vektor `X` die Punkte, an denen interpoliert werden soll.

Zeile die Zwischenwerte des Verfahrens auflistet.



Die folgende Aussage beweist nicht nur Satz 7.20, sondern sogar etwas mehr ...

Proposition 7.21 *(Eigenschaften von Aitken–Neville)*

1. *Die Abbildungen* $x \mapsto \boldsymbol{f}_k^j(x)$, $k = 0, \ldots, n-j$ *sind Polynome vom Grad* $\leq j$.
2. *Für* $j = 0, \ldots, n$ *und* $k = 0, \ldots, n-j$ *gilt:*

$$\boldsymbol{f}_k^j(x_m) = \boldsymbol{f}_m, \qquad m = k, \ldots, k+j, \tag{7.33}$$

insbesondere gilt also (7.32).

Beweis: Eigenschaft 1. folgt sofort aus (7.30), (7.31) und der Tatsache, daß $u_j(\cdot \mid I) \in \Pi_1$, $j = 1, 2$, für jedes $I \subset \mathbb{R}$.

(7.33) beweisen wir durch Induktion über j, wobei der Fall $j = 0$ genau der Initialisierungschritt im Algorithmus von Aitken-Neville ist. Sei also (7.33) für ein $j \geq 0$ bewiesen. Dann ist, nach (7.30) und der Induktionsannahme,

$$\boldsymbol{f}_k^{j+1}(x_k) = \underbrace{u_0\left(x_k \mid I_k^{j+1}\right)}_{=1} \underbrace{\boldsymbol{f}_k^j(x_k)}_{=\boldsymbol{f}_k} + \underbrace{u_1\left(x_k \mid I_k^{j+1}\right)}_{=0} \boldsymbol{f}_{k+1}^j(x_k) = \boldsymbol{f}_k,$$

sowie

$$\boldsymbol{f}_k^{j+1}(x_{k+j+1}) = \underbrace{u_0\left(x_{k+j+1} \mid I_k^{j+1}\right)}_{=0} \boldsymbol{f}_k^j(x_{k+j+1}) + \underbrace{u_1\left(x_k \mid I_k^{j+1}\right)}_{=1} \underbrace{\boldsymbol{f}_{k+1}^{j+1}(x_{k+j+1})}_{=\boldsymbol{f}_{k+j+1}} = \boldsymbol{f}_{k+j+1}.$$

Für $m = k+1, \ldots, k+j$ haben wir schließlich nach Induktionsannahme daß $\boldsymbol{f}_k^j(x_m) = \boldsymbol{f}_{k+1}^j(x_m) = \boldsymbol{f}_m$ und daher

$$\begin{aligned} \boldsymbol{f}_k^{j+1}(x_m) &= u_0\left(x_m \mid I_k^{j+1}\right) \underbrace{\boldsymbol{f}_k^j(x_m)}_{=\boldsymbol{f}_m} + u_1\left(x_m \mid I_k^{j+1}\right) \underbrace{\boldsymbol{f}_{k+1}^j(x_m)}_{=\boldsymbol{f}_m} \\ &= \underbrace{\left(u_0\left(x_m \mid I_k^{j+1}\right) + u_1\left(x_m \mid I_k^{j+1}\right)\right)}_{=1} \boldsymbol{f}_m = \boldsymbol{f}_m \end{aligned}$$

also gilt (7.33) auch für $j+1$, womit der Beweis beendet ist. □

Beweis von Satz 7.19: Die Existenz eines Interpolationspolynoms haben wir, wenn auch etwas umständlich, bewiesen. Die Eindeutigkeit folgt sofort aus der Tatsache, daß jedes Polynom vom Grad $\leq n$ mit $n+1$ Nullstellen das Nullpolynom sein muß: Sei $\boldsymbol{p} \in \Pi_n$ und sei (ohne Einschränkung) $\boldsymbol{a}_n \neq 0$. Nach dem Satz von Rolle hat dann die Ableitung $\boldsymbol{p}^{(j)}$ mindestens $n+1-j$ Nullstellen, also gibt es (mindestens) ein $x \in \mathbb{R}$, so daß

$$0 = \frac{\boldsymbol{p}^{(n)}(x)}{n!} = \boldsymbol{a}_n,$$

was einen Widerspruch liefert. □

7.3 Interpolation – der Newton–Ansatz

Der Algorithmus von Aitken–Neville ist eine feine Sache, wenn es darum geht, den Wert eines Interpolationspolynoms an einer Stelle x zu berechnen. Er hat aber auch seine Nachteile:

1. Wenn man das Interpolationspolynom formal manipulieren will (ableiten, zu einem anderen Polynom addieren und so), braucht man eine explizite Darstellung des Interpolationspolynoms.

2. Jede Auswertung mit Aitken–Neville hat einen Rechenaufwand von $O\left(n^2\right)$, was deutlich mehr als das $O(n)$ des Hornerschemas[68] ist. Muß man also ein Interpolationspolynom sehr oft auswerten, kann sich eine andere Darstellung lohnen.

Definition 7.22 *Es seien $x_0 < x_1 < \cdots < x_n$ und $\boldsymbol{f} = [\boldsymbol{f}_0, \ldots, \boldsymbol{f}_n] \in \mathbb{R}^{d\times n+1}$.*

1. *Der* Interpolationsoperator $L_n : \mathbb{R}^{d\times n+1} \to \Pi_n^d$ *ist definiert als das eindeutige Polynom $L_n\boldsymbol{f} \in \Pi_n^d$ mit der Eigenschaft*

$$L_n\boldsymbol{f}\left(x_j\right) = \boldsymbol{f}_j, \qquad j = 0, \ldots, n.$$

2. *Für $\boldsymbol{f} \in C\left(\mathbb{R}\right)^d$ können wir $L_n\boldsymbol{f}$ ganz einfach erweitern, indem wir*

$$\boldsymbol{f} \quad \textit{mit} \quad \left[\boldsymbol{f}\left(x_0\right), \ldots, \boldsymbol{f}\left(x_n\right)\right]$$

   identifizieren.

3. *Die* Newton–Darstellung *von $L_n\boldsymbol{f}$ hat die Form*

$$L_n\boldsymbol{f}(x) = \sum_{j=0}^{n} \lambda_j\left(\boldsymbol{f}\right)\ \left(x-x_0\right)\cdots\left(x-x_{j-1}\right),$$

   für passende Koeffizienten[69] *$\lambda_j(\boldsymbol{f}) \in \mathbb{R}^d$.*

[68] Der Aufwand von de Casteljau ist übrigens $O\left(n^2\right)$.
[69] Die wir natürlich noch bestimmen müssen.

Bemerkung 7.23 *(Newton–Darstellung)*

1. *Die Newton–Darstellung eines Interpolationspolynoms $L_n\boldsymbol{f}$ läßt sich mit dem verallgemeinerten Horner–Schema aus Korollar 7.7 recht (rückwärts–) stabil auswerten: da wir nur eine Subtraktion bei der Berechnung von $\ell_j(x)$ ausführen müssen, kommen wir mit $\eta = 1$ aus.*

2. *Die Newton–Darstellung erlaubt das einfache Hinzufügen von Punkten: Sind ein weiterer Punkt x_{n+1} und ein weiterer Wert $\boldsymbol{f}_{n+1}$ gegeben, dann ist*

$$L_{n+1}\boldsymbol{f}(x) = L_n\boldsymbol{f}(x) + \lambda_{n+1}(\boldsymbol{f})\,(x-x_0)\cdots(x-x_n)\,.$$

3. *Das "hinzugefügte" Polynom $(x-x_0)\cdots(x-x_n)$ hat (nicht zufällig) die Eigenschaft, das (bis auf Konstante) eindeutige Polynom kleinsten Grades zu sein, das an $x_0,\ldots,x_n$ verschwindet.*

Machen wir uns jetzt also an die Konstruktion der Koeffizienten $\lambda_j(\boldsymbol{f})$ in der Newton–Darstellung. Dazu bemerken wir zuerst, daß uns die Interpolationsbedingungen das Gleichungssystem

$$\sum_{j=0}^{n} \lambda_j(\boldsymbol{f})\ (x_k-x_0)\cdots(x_k-x_{j-1}) = \boldsymbol{f}_k, \qquad k=0,\ldots,n,$$

also

$$\begin{bmatrix} 1 & x_0-x_0 & (x_0-x_0)(x_0-x_1) & \ldots & \prod_{j=0}^{n-1}(x_0-x_j) \\ 1 & x_1-x_0 & (x_1-x_0)(x_1-x_1) & \ldots & \prod_{j=0}^{n-1}(x_1-x_j) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n-x_0 & (x_n-x_0)(x_n-x_1) & \ldots & \prod_{j=0}^{n-1}(x_n-x_j) \end{bmatrix} \begin{bmatrix} \lambda_0(\boldsymbol{f}) \\ \lambda_1(\boldsymbol{f}) \\ \vdots \\ \lambda_n(\boldsymbol{f}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_0 \\ \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{f}_n \end{bmatrix}.$$

Die Matrix auf der linke Seite ist eine untere Dreiecksmatrix, also ist

$$\begin{bmatrix} 1 & & & \\ 1 & * & & \\ \vdots & \vdots & \ddots & \\ 1 & * & \ldots & * \end{bmatrix} \begin{bmatrix} \lambda_0(\boldsymbol{f}) \\ \lambda_1(\boldsymbol{f}) \\ \vdots \\ \lambda_n(\boldsymbol{f}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_0 \\ \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{f}_n \end{bmatrix},$$

also hängt $\lambda_j(\boldsymbol{f})$ nur von $\boldsymbol{f}_0,\ldots,\boldsymbol{f}_j$ und $x_0,\ldots,x_j$ ab und zwar *linear* in $\boldsymbol{f}_0,\ldots,\boldsymbol{f}_j$.

Definition 7.24 *Die* dividierte Differenz $[x_0,\ldots,x_n]\,\boldsymbol{f}$ *zu einer Matrix $\boldsymbol{f}\in\mathbb{R}^{d\times n+1}$ oder einer Funktion $\boldsymbol{f}\in C(\mathbb{R})^d$ ist rekursiv definiert als*

$$[x_j]\,\boldsymbol{f} = \boldsymbol{f}_j = \boldsymbol{f}(x_j)\,, \qquad j=0,\ldots,n \tag{7.34}$$

$$[x_0,\ldots,x_{j+1}]\,\boldsymbol{f} = \frac{[x_0,\ldots,x_j]\,\boldsymbol{f} - [x_1,\ldots,x_{j+1}]\,\boldsymbol{f}}{x_0-x_{j+1}}, \qquad j=0,\ldots,n-1. \tag{7.35}$$

```
%%  DivDiff.m (Numerik I)
%%  -----------------------------------------------------
%%  Dividierte Differenzen (Vektorwertig)
%%  Eingabe:
%%    f    Matrix der Funktionswerte (Vektoren!)
%%    X    Knotenvektor

function y = DivDiff( f,X )
  n = length( X );

  for j = 1:n-1
    for k = 1:n-j
      f( :,k ) = ( f ( :,k ) - f ( :,k+1 ) ) / ( X( k ) - X( k+j ) );
    end
  end

  y = f( :,1 );
%endfunction
```

Programm 7.4 `DivDiff.m`: Berechnung der dividierten Differenzen über ein Dreiecksschema aus der Rekursionsformel (7.35).

```
%%  NewtonKoeff.m (Numerik 1)
%%  ------------------------------------------------------
%%  Koeffizienten des Interpolationspolynoms in der
%%  Newton-Darstellung (nicht optimal!)
%%  Eingabe:
%%    f    zu interpolierende Werte
%%    X    Interpolationspunkte

function F = NewtonKoeff( f,X )
  n = length( X );

  for j = 1:n
    F( :,j ) = DivDiff( f, X( 1:j ) );
  end
%endfunction
```

Programm 7.5 `NewtonKoeff.m`: Naive Berechnung des Koeffizientenvektors für das Interpolationspolynom, jede dividierte Differenz wird separat berechnet.

Satz 7.25 *Es seien $x_0, \ldots, x_n \in \mathbb{R}$ alle verschieden und $\boldsymbol{f} \in \mathbb{R}^{d\times n+1}$ oder $\boldsymbol{f} \in C(\mathbb{R})^d$. Dann ist*

$$L_n\boldsymbol{f}(x) = \sum_{j=0}^{n} [x_0, \ldots, x_j]\,\boldsymbol{f}\ \ (x - x_0)\cdots(x - x_{j-1})\,, \qquad x \in \mathbb{R}, \tag{7.36}$$

und außerdem

$$\boldsymbol{f}(x) - L_n\boldsymbol{f}(x) = (x - x_0)\cdots(x - x_n)\,[x, x_0, \ldots, x_n]\,\boldsymbol{f}. \tag{7.37}$$

Bevor wir uns mit einigen Konsequenzen aus der Newton–Darstellung des Interpolationspolynoms und natürlich dem Beweis von Satz 7.25 beschäftigen, sehen wir uns erst schnell an, wie man so ein Interpolationspolynom mit `Matlab` bestimmt.

Zuerst brauchen wir natürlich die Koeffizienten, also die dividierten Differenzen. Der naive Ansatz wäre also, wie in `NewtonKoeff.m` die dividierten Differenzen sukzessive zu bestimmen. Das ist aber hochgradig ineffizient: Auf dem Weg zu Berechnung von $[x_0, \ldots, x_j]\,\boldsymbol{f}$ brauchen wir ja alle dividierten Differenzen $[x_0, \ldots, x_k]\,\boldsymbol{f}$, $k < j$. Deswegen packen wir alles

```
%%  NewtonKoeff1.m (Numerik 1)
%%  ------------------------------------------------------
%%  Koeffizienten des Interpolationspolynoms in der
%%  Newton-Darstellung (schon besser)
%%  Eingabe:
%%    f    zu interpolierende Werte
%%    X    Interpolationspunkte

function F = NewtonKoeff1( f,X )
  n = length( X );
  F = f;

  for j = 2:n
    for k = n:-1:j
      F( :,k ) = ( F( :,k-1 ) - F( :,k ) ) / ( X( k-j+1 ) - X( k ) );
    end
  end
%endfunction
```

Programm 7.6 `NewtonKoeff1.m`: Effiziente Berechnung des Koeffizientenvektors mit Überschreiben. Beachte: Der Aufwand zur Berechnun aller dividierten Differenzen für das Interpolationspolynom ist geanu derselbe wir für die Berechnung des höchsten Koeffizienten $[x_0, \ldots, x_n]\, \boldsymbol{f}$.

in ein Schema mit Überschreiben:

$$
\begin{array}{|c|c|c|c|c|c|c|}
\hline
[x_0]\,\boldsymbol{f} & & [x_0]\,\boldsymbol{f} & \ldots & [x_0]\,\boldsymbol{f} & & [x_0\boldsymbol{f}] \\
 & \searrow & & & & & \\
[x_1]\,\boldsymbol{f} & \rightarrow & [x_0,x_1]\,\boldsymbol{f} & \ldots & [x_0,x_1]\,\boldsymbol{f} & & [x_0,x_1]\,\boldsymbol{f} \\
 & \searrow & & & & & \\
[x_2]\,\boldsymbol{f} & \rightarrow & [x_1,x_2]\,\boldsymbol{f} & \ldots & [x_0,x_1,x_2]\,\boldsymbol{f} & & [x_0,x_1,x_2]\,\boldsymbol{f} \\
 & \searrow & & & & & \\
\vdots & & \vdots & & \vdots & & \vdots \\
 & \searrow & & & & & \\
[x_{n-1}]\,\boldsymbol{f} & \rightarrow & [x_{n-2},x_{n-1}]\,\boldsymbol{f} & \ldots & [x_0,\ldots,x_{n-1}]\,\boldsymbol{f} & & [x_0,\ldots,x_{n-1}]\,\boldsymbol{f} \\
 & \searrow & & & & \searrow & \\
[x_n]\,\boldsymbol{f} & \rightarrow & [x_{n-1},x_n]\,\boldsymbol{f} & \ldots & [x_1,\ldots,x_n]\,\boldsymbol{f} & \rightarrow & [x_0,\ldots,x_n]\,\boldsymbol{f} \\
\hline
\end{array}
$$

Wegen des Überschreibens muß jede Spalte *von unten nach oben* abgearbeitet werden. Der Code hierzu findet sich in `NewtonKoeff1.m`. Sind also eine Datenmatrix `f` und ein Vektor von Interpolationspunkten `X` gegeben, so liefert uns der `Matlab`–Aufruf

```
F = NewtonKoeff1( f,X );
```

```
%%  NHorner.m (Numerik 1)
%%  ----------------------------------------------------
%%  Hornerschema bezueglich Newton-Basis
%%  Eingabe:
%%    f    Koeffizientenvektor
%%    X    Interpolationspunkte
%%    x    Stelle

function y = NHorner( f,X,x )
  n = length( X );
  y = f( :,n );

  for j = n-1:-1:1
    y = ( x - X( j ) ) * y + f( :,j );
  end
%endfunction
```

Programm 7.7 `NHorner.m`: Das Hornerschema für die Newtondarstellung eines Polynoms.

den gewünschten Kooeffizientenvektor. Um das Interpolationspolynom an der Stelle `x` auszuwerten, rufen wir schließlich mit

```
NHorner( F,X,x )
```

die Newton–Variante des Hornerschemas aus `NHorner.m` auf.

Jetzt aber zurück zur Theorie ...

Korollar 7.26 *Die dividierten Differenzen sind symmetrisch in den Argumenten: Für jede Permutation* $\sigma : \{0, \ldots, n\} \to \{0, \ldots, n\}$ *ist*

$$\left[x_{\sigma(0)}, \ldots, x_{\sigma(n)}\right] \boldsymbol{f} = [x_0, \ldots, x_n]\, \boldsymbol{f}. \tag{7.38}$$

Beweis: Nach (7.36) ist

$$L_n \boldsymbol{f}(x) = [x_0, \ldots, x_n]\, \boldsymbol{f}\; x^n + q(x), \qquad q \in \Pi_{n-1},$$

und da $L_n \boldsymbol{f}$ invariant unter Vertauschung der Interpolationspunkte (bei gleichzeitiger Vertauschung der zu interpolierenden Werte) ist, folgt die Behauptung. □

Das folgende Resultat zeigt uns, was der Newton Ansatz algorithmisch macht: In jedem Schritt wird versucht, durch geeignete Interpolation des Fehlers[70] die Genauigkeit des Interpolanten Schritt für Schritt zu erhöhen.

[70] Ohne die vorher bereits erzielten Interpolationseigenschaften zu zerstören.

Korollar 7.27 *Es seien $x_0, \ldots, x_n \in \mathbb{R}$ alle verschieden und $\boldsymbol{f} \in \mathbb{R}^{d\times n+1}$ oder $\boldsymbol{f} \in C(\mathbb{R})^d$. Dann ist*[71]

$$L_n\boldsymbol{f}(x) = \sum_{j=0}^{n} \left(\boldsymbol{f} - L_{j-1}\boldsymbol{f}\right)(x_j)\ \frac{(x-x_0)\cdots(x-x_{j-1})}{(x_j-x_0)\cdots(x_j-x_{j-1})}. \tag{7.39}$$

Beweis: Nach (7.37) mit $n = j-1$ und $x = x_j$ ist

$$[x_0, \ldots, x_j]\,\boldsymbol{f} = \frac{\left(\boldsymbol{f} - L_{j-1}\boldsymbol{f}\right)(x_j)}{(x_j-x_0)\cdots(x_j-x_{j-1})}, \qquad j = 0, \ldots, n.$$

□

Beweis von Satz 7.25: Wir beweisen (7.36) und (7.37) simultan duch Induktion über n.
Für $n = 0$ ist $L_0\boldsymbol{f}(x) = \boldsymbol{f}_0 = \boldsymbol{f}(x_0)$ und

$$\boldsymbol{f}(x) - L_0\boldsymbol{f}(x) = \boldsymbol{f}(x) - \boldsymbol{f}(x_0) = (x-x_0)\ \frac{\boldsymbol{f}(x)-\boldsymbol{f}(x_0)}{(x-x_0)} = (x-x_0)\ [x_0, x]\,\boldsymbol{f}.$$

Seien also (7.36) und (7.37) für ein $n \geq 0$ bewiesen. Wir setzen

$$\begin{aligned} P_{n+1}\boldsymbol{f}(x) &:= \sum_{j=0}^{n+1} [x_0, \ldots, x_j]\,\boldsymbol{f}\ (x-x_0)\cdots(x-x_{j-1}) \\ &= P_n\boldsymbol{f}(x) + [x_0, \ldots, x_{n+1}]\,\boldsymbol{f}\ (x-x_0)\cdots(x-x_n)\,. \end{aligned}$$

Nach der Induktionsannahme erhalten wir aus (7.36) und (7.39)[72], daß

$$P_{n+1}\boldsymbol{f}(x) = L_n\boldsymbol{f}(x) + \left(\boldsymbol{f} - L_n\boldsymbol{f}\right)(x_{n+1})\ \frac{(x-x_0)\cdots(x-x_n)}{(x_{n+1}-x_0)\cdots(x_{n+1}-x_n)}.$$

Damit ist, für $j = 0, \ldots, n$,

$$P_{n+1}\boldsymbol{f}(x_j) = L_n\boldsymbol{f}(x_j) = \boldsymbol{f}_j$$

und außerdem

$$P_{n+1}\boldsymbol{f}(x_{n+1}) = L_n\boldsymbol{f}(x_{n+1}) + \boldsymbol{f}(x_{n+1}) - L_n\boldsymbol{f}(x_{n+1}) = \boldsymbol{f}(x_{n+1}) = \boldsymbol{f}_{n+1},$$

also $P_{n+1}\boldsymbol{f} = L_{n+1}\boldsymbol{f}$ und (7.36) gilt auch für $n+1$. Andererseits ist mit dem gerade bewiesenen und nochmals der Induktionsannahme

$$\begin{aligned} \boldsymbol{f}(x) - L_{n+1}\boldsymbol{f}(x) &= \left(\boldsymbol{f}(x) - L_n\boldsymbol{f}(x)\right) - \left(L_{n+1}\boldsymbol{f}(x) - L_n\boldsymbol{f}(x)\right) \\ &= (x-x_0)\cdots(x-x_n)\ [x, x_0, \ldots, x_n]\,\boldsymbol{f} - (x-x_0)\cdots(x-x_n)\ [x_0, \ldots, x_n, x_{n+1}]\,\boldsymbol{f} \\ &= (x-x_0)\cdots(x-x_{n+1})\ \frac{[x, x_0, \ldots, x_n]\,\boldsymbol{f} - [x_0, \ldots, x_n, x_{n+1}]\,\boldsymbol{f}}{x - x_{n+1}} \\ &= (x-x_0)\cdots(x-x_{n+1})\ [x, x_0, \ldots, x_{n+1}]\,\boldsymbol{f}, \end{aligned}$$

[71] Mit $L_{-1}(x) = 0$.
[72] Da (7.39) direkt aus (7.37) folgt ist dessen Verwendung zulässig!

nach der Rekursionsformel (7.35). □

Übung 7.3 Zeigen Sie:

$$p \in \Pi_n \qquad \Longleftrightarrow \qquad [x_0, \ldots, x_k]\, p = 0, \quad x_0 < \cdots < x_k, \quad k > n.$$

Schließlich noch eine Aussage über den Interpolationsfehler für hinreichend oft differenzierbare Funktionen.

Satz 7.28 *Seien $x_0, \ldots, x_n$ alle verschieden und $f \in C^{n+1}(\mathbb{R})$. Dann gibt es für jedes $x \in \mathbb{R}$ ein $\xi \in [x_0, \ldots, x_n, x]$, der konvexen Hülle von $x_0, \ldots, x_n, x$[73], so daß*

$$f(x) - L_n f(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\ (x - x_0) \cdots (x - x_n)\,. \tag{7.40}$$

Dieses Resultat gilt nur für $d = 1$, also skalare Funktionen. Für Vektorfelder $\boldsymbol{f}$ wird es im Normalfall für jede Komponente f_j, $j = 1, \ldots, d$, ein "eigenes" ξ_j geben, so daß (7.40) gilt.

Auf alle Fälle liefern uns die Fehlerformeln (7.40) und (7.37) auch noch eine Aussage über die dividierten Differenzen differenzierbarer Funktionen.

Korollar 7.29 *Es sei $f \in C^n(\mathbb{R})$. Zu beliebigen verschiedenen Punkten $x_0, \ldots, x_n \in \mathbb{R}$ gibt es ein $\xi \in [x_0, \ldots, x_n]$, so daß*

$$[x_0, \ldots, x_n]\, f = \frac{f^{(n)}(\xi)}{n!}.$$

Beweis von Satz 7.28: Für $x \notin \{x_0, \ldots, x_n\}$ setzen wir

$$\alpha = \frac{f(x) - L_n f(x)}{(x - x_0) \cdots (x - x_n)},$$

definieren

$$g(y) = f(y) - L_n f(y) - \alpha\,(y - x_0) \cdots (y - x_n)\,, \qquad y \in \mathbb{R},$$

und bemerken, daß $g\,(x_j) = 0$, $j = 0, \ldots, n$, sowie $g(x) = 0$. Nach dem Satz von Rolle hat also $g^{(j)}$ mindestens $n + 2 - j$ Nullstellen im Intervall $[x_0, \ldots, x_n, x]$, $j = 0, \ldots, n+1$, es gibt also mindestens ein $\xi \in [x_0, \ldots, x_n, x]$, so daß

$$\begin{aligned} 0 \;&=\; g_\alpha^{(n+1)}(\xi) = f^{(n+1)}(\xi) - \underbrace{\left(L_n f(x)\right)^{(n+1)}}_{=0} - \alpha\, \underbrace{\frac{d^{n+1}}{dx^{n+1}}\,(x - x_0) \cdots (x - x_n)}_{=(n+1)!} \\ &=\; f^{(n+1)}(\xi) - \alpha\ (n+1)!, \end{aligned}$$

also ist $\alpha = f^{(n+1)}(\xi)/(n+1)!$. □

[73] Alternativ: das kleinste Intervall, das $x_0, \ldots, x_n$ und x enthält.

7.4 Approximationsgüte von Interpolationspolynomen

Wir beginnen mit einer (trivialen) Folgerung aus Satz 7.28.

Korollar 7.30 *Für $f \in C^{n+1}(\mathbb{R})$ und $x_0, \ldots, x_n \in I \subset \mathbb{R}$ ist*

$$\|f - L_n f\|_I = \max_{x \in I} |f(x) - L_n f(x)| \leq \frac{\left\|f^{(n+1)}\right\|_I}{(n+1)!} \max_{x \in I} |x - x_0| \cdots |x - x_n| \,. \tag{7.41}$$

Wollen wir nun, daß der Interpolant $L_n f$ die Funktion f gut approximiert[74], dann haben wir nur eine Wahl: Wir müssen die Interpolationspunkte $x_0, \ldots, x_n$ so wählen, daß der Ausdruck

$$\max_{x \in I} |x - x_0| \cdots |x - x_n| = \|(\cdot - x_0) \cdots (\cdot - x_n)\|_I$$

möglichst klein wird. Wenn wir uns auf $I = [-1, 1]$ beschränken, können wir diese Punkte *explizit* bestimmen, für beliebige Intervalle a, b verwendet man dann die Transformation

$$x \mapsto a + \frac{x+1}{2}\,(b - a),$$

die $[-1, 1]$ linear auf $[a, b]$ abbildet.

Bemerkung 7.31 *Es bezeichne $\mathscr{P}_n \subset \Pi_n$ die Menge aller Polynome vom Grad n mit* monomialem Leitterm*:*

$$p \in \mathscr{P}_n \qquad \Longleftrightarrow \qquad p(x) = x^n + q(x), \quad q \in \Pi_{n-1}.$$

Ist also p^* das Polynom, für das

$$\|p^*\|_I = \min_{p \in \mathscr{P}_n} \|p\|_I$$

gilt, dann sind die Nullstellen[75] von p^* optimale Interpolationspunkte.
Unser Ziel wird es nun sein, für alle $n \in \mathbb{N}_0$ so ein optimales p^* zu finden.

Definition 7.32 *Das n–te* Tschebyscheff–Polynom[76] *T_n, $n \in \mathbb{N}$, ist definiert als*

$$T_n(x) = \cos\left(n \arccos x\right).$$

Und, kaum zu glauben, aber wahr, die Tschebyscheffpolynome sind die Lösung unseres Problems.

[74] Das wäre ja wohl das, was man als "gute Rekonstruktion" bezeichnen würde.
[75] Man sollte aber schon nachweisen, daß diese Nullstellen auch alle einfach und rell sind!
[76] Es gibt eine Vielzahl von Transkriptionen des Namen "Tschebyscheff", z.B. auch Chebychev, Chebychov.

Satz 7.33 *Die Polynome $2^{1-n}T_n$ gehören zu $\mathscr{P}_n$, $n \geq 1$[77], und es gilt*

$$\left\|2^{1-n}T_n\right\|_{[-1,1]} = \min_{p\in\mathscr{P}_n} \| \, p \, \|_{[-1,1]} \, . \tag{7.42}$$

Die Nullstellen von T_{n+1}[78], $n \in \mathbb{N}_0$, sind alle einfach und rell und damit optimale Interpolationspunkte für Π_n.

Korollar 7.34 *Sei L_n^* der Interpolationsoperator an den Nullstellen von T_{n+1}. Dann ist*

$$\|f - L_n^* f\|_{[-1,1]} \leq \frac{\left\|f^{(n+1)}\right\|_{[-1,1]}}{2^n \, (n+1)!}. \tag{7.43}$$

Um einzusehen, daß tatsächlich $T_n \in \Pi_n$ ist, brauchen wir etwas mehr Information über die Tschebyscheff–Polynome.

Lemma 7.35 *Die Tschebyscheff–Polynome T_n, $n \in \mathbb{N}$, erfüllen die Rekursionsformel*

$$\begin{aligned} T_0(x) &= 1, && (7.44)\\ T_1(x) &= x, && (7.45)\\ T_{n+1}(x) &= 2x \, T_n(x) - T_{n-1}(x), \qquad n \in \mathbb{N}. && (7.46) \end{aligned}$$

Beweis: Die Anfangsbedingungen (7.44) und (7.44) folgen trivialerweise aus der Definition (und $\cos 0 = 1$).
Wir setzen $x = \cos\theta$, d.h., $\theta = \arccos x$, also $T_n(x) = \cos n\theta$. Durch Addition der Additionstheoreme

$$\begin{aligned} \cos(n+1)\theta &= \cos n\theta \;\cos\theta + \sin n\theta \;\sin\theta \\ \cos(n-1)\theta &= \cos n\theta \;\cos\theta - \sin n\theta \;\sin\theta \end{aligned}$$

erhalten wir, daß

$$\underbrace{\cos(n+1)\theta}_{=T_{n+1}(x)} + \underbrace{\cos(n-1)\theta}_{=T_{n-1}(x)} = 2\underbrace{\cos\theta}_{=x}\underbrace{\cos n\theta}_{=T_n(x)},$$

also (7.46). □

Als nächstes Information über die Nullstellen.

Proposition 7.36 *Die Nullstellen von T_n sind*

$$x_j = \cos\frac{2j-1}{2n}\pi, \qquad j = 1, \ldots, n. \tag{7.47}$$

[77] Der Fall $n = 0$ verlangt den Faktor 1 und nicht $\frac{1}{2}$, siehe (7.44); andererseits hat T_0 auch keine Nullstelle und ist daher ohnehin zur Bestimmung von Interpolationsknoten ungeeignet.

[78] Siehe (7.47).

```
%%  TschebNodes.m (Numerik 1)
%%  ------------------------------------------------------
%%  Tschebyscheffknoten
%%  Eingabe:
%%    n    % Knoten

function X = TschebNodes( n )
  X = zeros( 1,n+1 );

  for j=1:n+1
    X( j ) = cos( (2*j-1)/(2*n+2)*pi );
  end
%endfunction
```

Programm 7.8 `TschebNodes.m`: Knotenvektor der Tschbyscheffpolynome.

Außerdem sind die Nullstellen von T_n'

$$x_j' = \cos\frac{j}{n}\pi, \qquad j = 1, \ldots, n-1, \tag{7.48}$$

und es ist[79]*, mit* $x_0' = -1$*,* $x_n' = 1$*,*

$$T_n\left(x_j'\right) = (-1)^{n+j}, \qquad j = 0, \ldots, n. \tag{7.49}$$

Also ist insbesondere $\|T_n\|_{[-1,1]} = 1$*.*

Beweis: Es ist

$$T_n\left(x_j\right) = \cos\left(n \arccos\, \cos\frac{2j-1}{2n}\pi\right) = \cos\frac{2j-1}{2}\pi = 0, \qquad j = 1, \ldots, n,$$

und mehr als n Nullstellen kann T_n ja nun mal nicht haben, denn $T_n(x) = 2^{n-1}x^n + \cdots \not\equiv 0$ nach (7.46). Deswegen sind die Nullstellen auch alle einfach. Außerdem ist

$$T_n'(x) = \frac{n}{\sqrt{1-x^2}}\, \sin\left(n \arccos x\right)$$

und da $x_j' \neq -1, 1$ und $\sin j\pi = 0$, $j \in \mathbb{N}_0$, ist auch $T_n'\left(x_j\right) = 0$, $j = 1, \ldots, n-1$. Also sind die potentiellen Extrempunkte von T_n auf dem Intervall $[-1, 1]$ die Punkte

$$-1, x_1', \ldots, x_{n-1}', 1.$$

[79] Wenn man genau ist: für $n \geq 1$.

Aus (7.46) erhalten wir sofort, daß $T_n(-1) = (-1)^n$ und $T_n(1) = 1$ und darüberhinaus ist

$$T_n\left(x'_j\right) = \cos\frac{j}{n}\pi = (-1)^j,$$

woraus (7.48) folgt. □

Beweis von Satz 7.33: Daß T_n einfache, reelle Nullstellen hat, wurde ja in Proposition 7.36 bewiesen. Daß $2^{1-n}T_n \in \mathscr{P}_n$ folgt aus der Rekursionsformel (7.46). Außerdem wissen wir, daß $\|2^{1-n}T_n\|_{[-1,1]} = 2^{1-n}$. Nehmen wir also an, es gäbe ein $p \in \mathscr{P}_n$, so daß

$$\| p \|_{[-1,1]} < 2^{1-n}$$

und setzen wir $q = 2^{1-n}T_n - p$. Da $2^{1-n}T_n, p \in \mathscr{P}_n$, ist $q \in \Pi_{n-1}$. An den Stellen x'_j, $j = 0, \ldots, n$[80], ist dann

$$q\left(x'_j\right) = (-1)^j 2^{1-n} - p\left(x_j\right) =: (-1)^j \mu_j, \qquad j = 0, \ldots, n,$$

wobei $\mu_j > 0$, $j = 0, \ldots, n$. Damit muß aber q zwischen x'_j und x'_{j+1}, $j = 0, \ldots, n-1$, mindestens eine Nullstelle haben, also hat $q \in \Pi_{n-1}$ mindestens n Nullstellen und ist damit das Nullpolynom. Dann führt aber $2^{1-n}T_n = p$ und somit

$$2^{1-n} > \| p \|_{[-1,1]} = \left\|2^{1-n}T_n\right\|_{[-1,1]} = 2^{1-n}$$

zum Widerspruch. □

Tatsächlich liefern die Tschebyscheffknoten *entscheidend* bessere Interpolanten als beispielsweise gleichverteilte Knoten.

Beispiel 7.37 *Wir betrachten die geschlossene polynomiale* Kurve*, die durch die Punkte (*`Matlab`*–Vektor* `f`*)*

$$\boldsymbol{f} = \begin{bmatrix} 0 & \frac{1}{4} & \frac{1}{2} & \frac{3}{4} & 1 & 1 & 1 & 1 & 1 & \frac{3}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{3}{4} & 1 & 1 & 1 & 1 & 1 & \frac{3}{4} & \frac{1}{2} & \frac{1}{4} & 0 \end{bmatrix}$$

auf dem Rand des Einheitsquadrats geht (siehe Abb. 7.5). Als Parameterbereich wählen wir $[-1, 1]$ *und interpolieren einmal an gleichverteilen Punkten mit*

```
auxPlotNewton( f, (-1:1/8:1), (-1:.01:1) )
```

und einmal an den Tschebyscheffknoten:

```
auxPlotNewton( f, TschebNodes( 16 ), (-1:.01:1) )
```

Die Ergebnisse sind, wie Abb. 7.6 zeigt, sehr unterschiedlich.

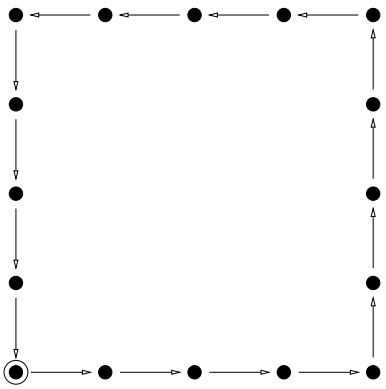


Abbildung 7.5: Interpolationsdaten für die Kurve. Frei ist noch die Wahl der zugehörigen Parameter (=Knoten).

7.5 Die schlechte Nachricht

Da man also so toll, einfach und halbwegs stabil mit Polynomen modellieren (Bézierkurven) und rekonstruieren (Interpolation) kann, werden es ja wohl alle CAD–Systeme und numerische Anwendungen verwenden, oder? Die Antwort ist ja und nein: Bézierkurven sind in den meisten Zeichenprogrammen tatsächlich als ein Mittel zum Freihandzeichnen enthalten und polynomiale Interpolation versteckt sich tatsächlich hinter einigen numerischen Verfahren. Aber:

- Komplexität und Instabilität wachsen mit dem Grad des Polynoms an. Beispielsweise dauert die Auswertung einer Bézierkurve mit 100 Kontrollpunkten doch schon ein bißchen.
- Polynome sind *global*. Die Änderung eines einzigen Kontrollpunkts ändert die Kurve überall.
- Interpolationspolynome haben einen krankhaften Drang, zu oszillieren, wie Abb. 7.7 zeigt. Allerdings sieht man dort auch, daß es so richtig schlimm erst außerhalb der konvexen Hülle der Interpolationspunkte wird, aber auch in deren Inenren wird die Oszillation für wachsenden Grad immer heftiger

Noch schlimmer ist aber die Empfindlichkeit polynomialer Interpolation gegen Störungen. Dazu ein Beispiel.

Beispiel 7.38 *Wir betrachten die Parabel*[81] $p(x) = x(1-x)$ *und interpolieren an den* 21 *gleichverteilten Punkten* $0, \frac{1}{20}, \ldots, 1$. *Das Interpolationspolynom ist auch kaum von der Parabel selbst zu unterscheiden (Abb. 7.8). Stören wir aber jeden Datenwert zufällig um* $< 0.1\%$, *dann erhalten wir bereits eine ziemlich unögliche "Flugbahn" (Abb. 7.8), stören wir gar den Scheitelpunkt um* 1%, *so wird der Interpolant endgültig wüst, siehe Abb. 7.9. Auf der anderen Seite*

[80] Wir nehmen also die Grenzen ± 1 mit dazu.

[81] Ein Schuft ist, wer an Geschoßbahnen dabei denkt.

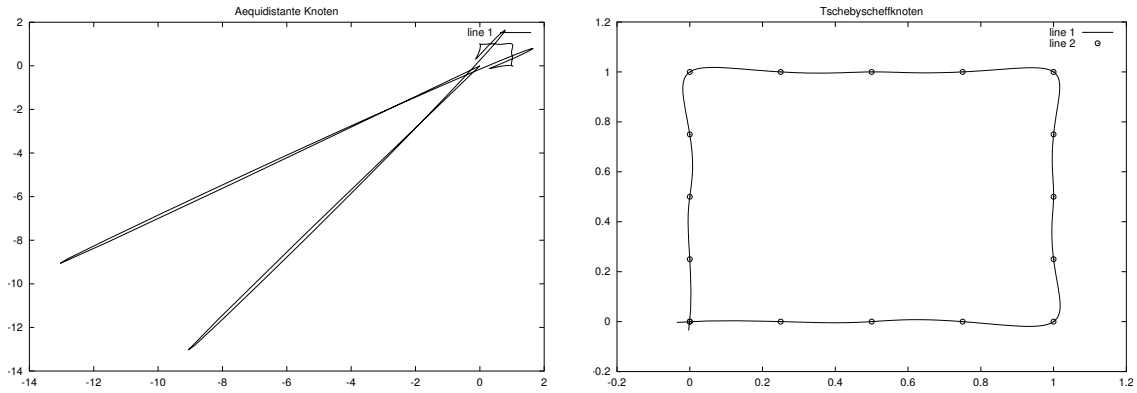


Abbildung 7.6: Interpolation der “Quadratkurve”, einmal mit gleichverteilten Parameterwerten für die Interpolationspunkte, einmal unter Verwendung der Tschebyscheffknoten.

zeigt Abb. 7.10, daß sogar wesentlich heftigere Störungen $\sim 1\%$ weggesteckt werden, wenn die Tschebyscheffknoten verwendet werden.

Das Problem ist nur leider, daß man sich in der Praxis die Interpolationspunkte nicht aussuchen kann, sondern daß diese beispielsweise von den physikalischen Eigenschaften des Meßprozesses vorgegeben werden. Außerdem ist das Hinzufügen von Punkten nicht mehr so einfach: die Polynome T_n und T_{n+1} haben keine gemeinsamen Nullstellen[82]!

[82] Wohl aber T_n und T_{2n} – was auch in manchen Verfahren ausgenutzt wird.

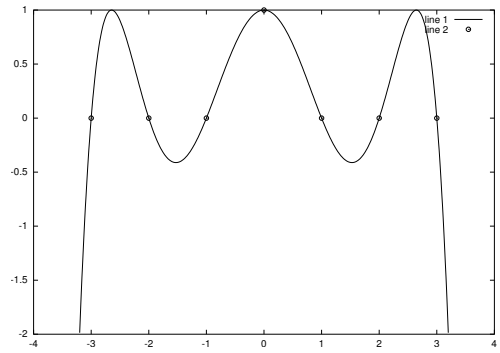


Abbildung 7.7: Interpolationspolynom, das den Wert 0 an $\pm 1, \pm 2$ und 1 an der Stelle 0 interpoliert. Nicht ganz das, was man gerne hätte.

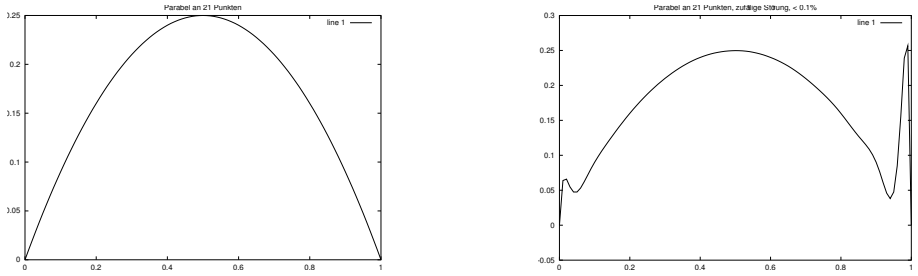


Abbildung 7.8: Interpolanten vom Grad 20 für die Originaldaten und leicht ($< 0.1\%$) gestörte Werte.

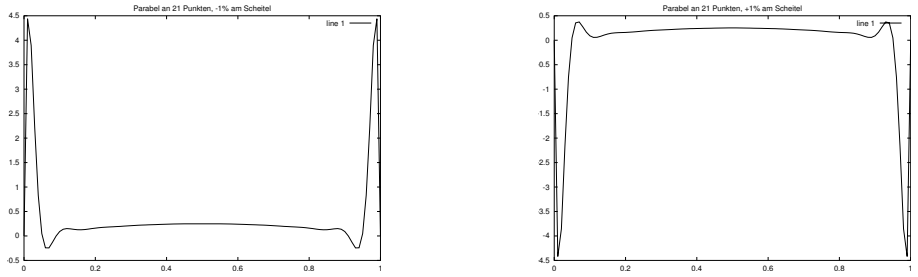


Abbildung 7.9: Interpolanten vom Grad 20 für Störung des Scheitelpunkts um $\pm 1\%$.

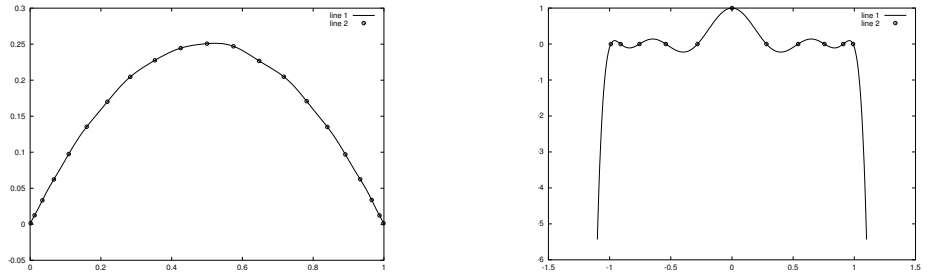


Abbildung 7.10: Zufällige Sörungen an *allen* Interpolationspunkten um $< 1\%$ tun bei Verwendung der Tschebyscheffknoten bei weitem nicht so weh. Auch die Oszillation ist nicht so groß, aber dafür wird das Gefälle außerhalb von $[-1, 1]$ umso steiler.

Im Wort Gelehrter steckt nur der Begriff, daß man ihn vieles gelehrt, aber nicht, daß er auch etwas gelernt hat.

Lichtenberg

8 Splines

Um die Probleme polynomialer Kurven, insbesondere Interpolanten, also

- Globalität,
- Oszillation,
- hohe Komplexität,

zu umgehen, brauchen wir eine Kurvenkonstruktion, bei der der Wert der Kurve nur *lokal* von einem Teil der Koeffizienten abhängt. Um die *Lokalität* beschreiben und (flexibel) kontrollieren zu können brauchen wir ein bißchen Terminologie.

Definition 8.1 *Seien* $n \geq m \geq 0$[83].

1. *Eine Punktmenge* $T = T_{m,n} = \{t_0, \ldots, t_{n+m+1}\} \subset \mathbb{R}$ *heißt* Knotenfolge *der Ordnung* m*, falls*

$$t_0 \leq t_1 \leq \cdots \leq t_n \leq \cdots \leq t_{n+m+1} \tag{8.1}$$

   und

$$t_j < t_{j+m+1}, \qquad j = 0, \ldots, n. \tag{8.2}$$

2. *Die* Knoten $t_0, \ldots, t_m$ *bzw.* $t_{n+1}, \ldots, t_{n+m+1}$ *heißen* linke *bzw.* rechte Randknoten*, die Knoten* $t_{m+1}, \ldots, t_n$ *heißen* innere Knoten*.*

3. *Dasjenige* $\mu > 0$[84]*, für das*

$$t_{j-1} < t_j = \cdots = t_{j+\mu-1} < t_{j+\mu}$$

   gilt, heißt Vielfachheit *des Knotens* $t_j = \cdots = t_{j+\mu-1}$*.*

4. *Das Intervall* I_j^k *ist definiert als*

$$I_j^k = [t_j, t_{j+k}], j = 0, \ldots, n, \quad k = 1, \ldots, m.$$

[83] Das ist *formal* ausreichend. Die "Vorstellung" ist allerdings, daß $n \gg m$ und m *sehr* moderat (etwa $m = 3$) ist.

[84] Wie "μultiplicity"

8.1 Der Algorithmus von de Boor

Als nächstes definieren wir uns auf *algorithmische Weise* die Kurven, die uns interessieren werden.

Algorithmus 8.2 *(de Boor)*
Gegeben: *Knotenfolge* $T = T_{m,n}$*, Kontrollpunkte* $\boldsymbol{d}_0, \ldots, \boldsymbol{d}_n \in \mathbb{R}^d$ *und* $x \in [t_m, \ldots, t_{n+1}]$.

1. *Bestimme* $r \in \{m, \ldots, n\}$ *so daß* $x \in [t_r, t_{r+1})$.[85]
2. *Setze*
$$\boldsymbol{d}_k^0(x) = \boldsymbol{d}_k, \qquad k = r-m, \ldots, r.$$
3. *Für* $j = 1, \ldots, m$ *berechne*
$$\boldsymbol{d}_k^j(x) = u_0\left(x|I_k^{m-j+1}\right)\boldsymbol{d}_{k-1}^{j-1}(x) + u_1\left(x|I_k^{m-j+1}\right)\boldsymbol{d}_k^{j-1}(x), \qquad k = r-m+j, \ldots, r. \tag{8.3}$$
4. *Ergebnis:* $N_{m,T}\boldsymbol{d}(x) := \boldsymbol{d}_r^m(x)$.

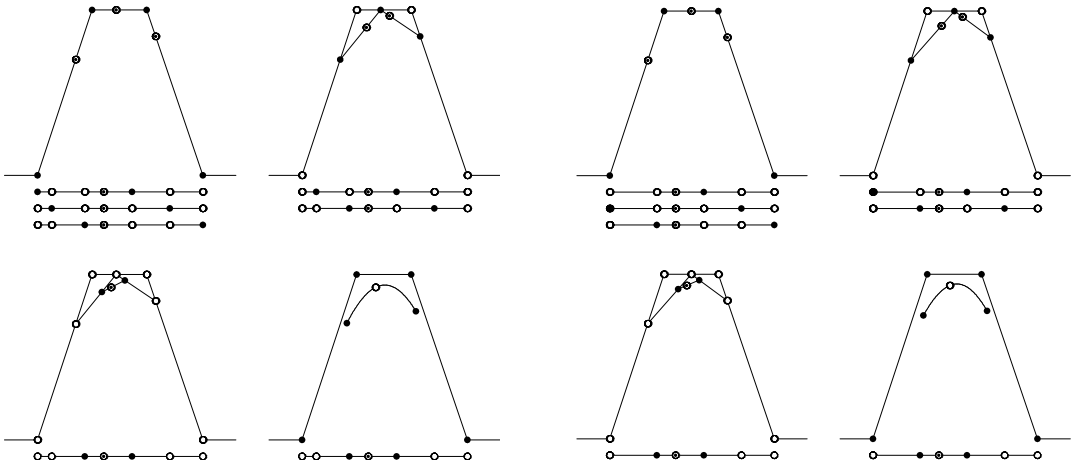


Abbildung 8.1: Der Algorithmus von de Boor für $m = 3$ und einen einfachen sowie einen doppelten Knoten.

Die Abbildung 8.1 zeigt, was beim Algorithmus von de Boor passiert: im j–ten Schritt werden alle Intervalle abgearbeitet, die $m - j + 2$ Knoten und den Punkt x enthalten. Der Index r aus Schritt 1 im obigen Algorithmus ist eindeutig, solange x nicht gerade mit einem Knoten übereinstimmt.

[85] Ein solches r existiert *nicht*, wenn $t_{n+1} = \cdot = t_{n+m+1}$ ein $(m+1)$–facher Randknoten und $x = t_{n+1}$ ist. Wir können uns aber dadurch behelfen, daß wir uns in diesem Fall irgendein $t_{n+m+2} > t_{n+m+1}$ dazuerfinden und mit $r = n + m$ weitermachen.

```
%%  deBoor.m (Numerik 1)
%%  ---------------------------------------------------------
%%  Algorithmus von de Boor
%%  Eingabe:
%%    d    Kontrollpunkte (vektorwertig!)
%%    T    Knotenvektor
%%    x    Stelle

function y = deBoor( d,T,x )
  n = length( d );
  m = length( T ) - n - 1;

  % Finde Knoten

  r = m+1;
  while (r < n ) & ( x >= T( r+1 ) )
    r = r + 1;
  end

  % Behebe Problem mit rechtem Randpunkt bei (m+1)-fachen Knoten

  if ( r == n )
    y = d( :,n );
  end

  % Schleife (mit Ueberschreiben)

  for j = 1:m
    for k = r:-1:r-m+j
      u = ( T( k+m-j+1 ) - x ) / ( T( k+m-j+1) - T( k ) );
      d( :,k ) = u * d( :,k-1) + (1-u) * d( :,k );
    end
  end

  y = d( :,r );
%endfunction
```

Programm 8.1 `deBoor.m`: Der Algorithmus von de Boor mit Überschreiben des Koeffizientenvektors.

Bemerkung 8.3 *Der Index r, der durch die Forderung $x \in [t_r, t_{r+1})$ in Algorithmus 8.2 festgelegt ist, beeinflußt* nicht *die Berechnungsformel, sondern lediglich,* welche *Kontrollpunkte zur Berechnung zu verwenden sind. Man könnte auch den folgenden, modifizierten Algorithmus verwenden:*

1. *Setze*
$$\boldsymbol{d}_k^0(x) = \boldsymbol{d}_k, \qquad k = 0, \ldots, n.$$

2. *Für $j = 1, \ldots, m$ berechne*
$$\boldsymbol{d}_k^j(x) = u_0\left(x|I_k^{m-j+1}\right)\boldsymbol{d}_{k-1}^{j-1}(x) + u_1\left(x|I_k^{m-j+1}\right)\boldsymbol{d}_k^{j-1}(x), \qquad k = j, \ldots, n. \quad (8.4)$$

3. *Wähle die r–te Komponente aus dem Ergebnisvektor $(\boldsymbol{d}_k^m(x) : k = m, \ldots, n)$.*

Aber: *der Rechenaufwand dieses Verfahrens ist* wesentlich *höher, nämlich $O(nm)$ Operationen im Gegensatz zu $O(m^2)$ im Algorithmus von de Boor. Und normalerweise ist $n \gg m$!*

Definition 8.4 *Die Kurve $N_{m,T}\boldsymbol{d} : [t_m, t_{n+1}]$ heißt* Splinekurve *der Ordnung m*[86].

Als nächstes ein paar einfache Eigenschaften der Splinekurve, die sofort aus dem Algorithmus folgen:

1. Die Splinekurve ist nur auf dem Intervall $[t_m, t_{n+1}]$, das von den beiden *innersten* Randknoten gebildet wird, definiert. Die anderen Randknoten beeinflussen aber natürlich den Wert der Splinekurve, zumindest in der Nähe des Randes[87].

2. Es ist selbstverständlich nicht verboten, die Randknoten alle identisch, also $(m+1)$–fach, zu wählen. Im Gegenteil, dieser Fall hat eine nette Eigenschaft: für $x = t_m$ (also $r = m$) ist dann für $j = 1, \ldots, m$ und $k = r - m + j, \ldots, r = j, \ldots, m$[88] das Intervall
$$I_k^{m-j+1} = [t_k, t_{k+m-j+1}] = [t_m, t_{k+m-j+1}]$$
zu betrachten[89] und somit ist
$$u_0\left(x|I_k^{m-j+1}\right) = u_0\left(t_m \mid [t_m, t_{k+m-j+1}]\right) = 1, \qquad k = r-m+j, \ldots, r, \quad j = 1, \ldots, m,$$
also auch
$$\boldsymbol{d}_{r-m+j}^j(x) = \boldsymbol{d}_j^j(x) = \boldsymbol{d}_0,$$
insbesondere ist also $\boldsymbol{d}_0 = \boldsymbol{d}_m^m(t_m) = N_{m,T}(t_m)$. Da dasselbe für den rechten Rand gemacht werden kann, besitzen also die Splinekurven die Eigenschaft der *Endpunktinterpolation*, wenn wir $(m+1)$–fache Randknoten verwenden.

3. Dies gilt natürlich nicht nur für Randknoten: An jedem m–fachen inneren Knoten $t_j = \cdots = t_{j+m-1}$ wird der entsprechende Koeffizient $\boldsymbol{d}_{j-m}$ interpoliert, d.h. $N_{m,T}\boldsymbol{d}(t_j) =$

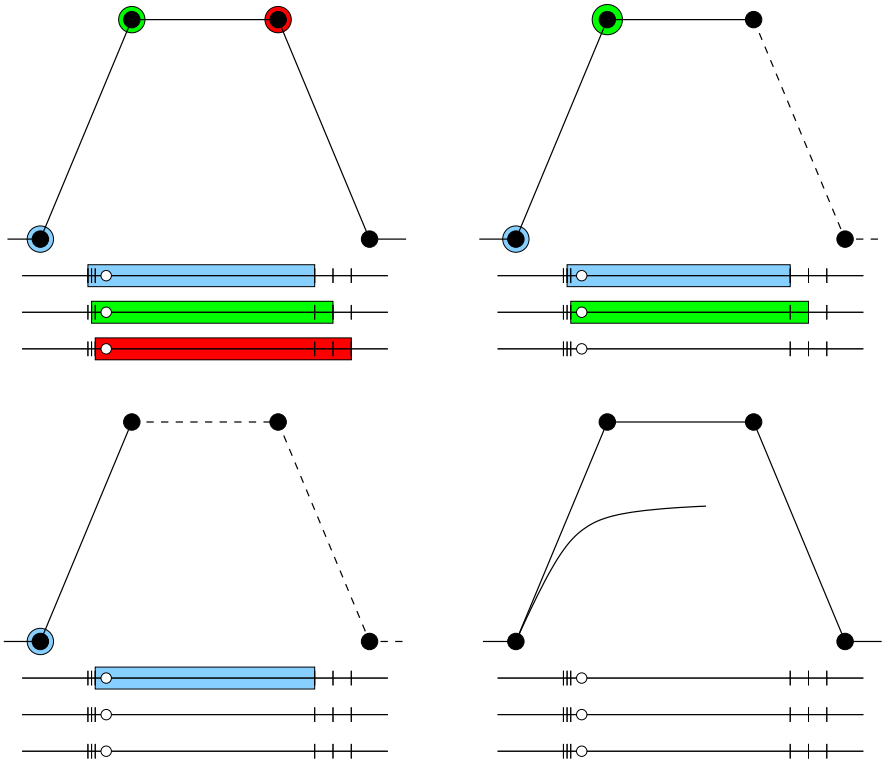


Abbildung 8.2: Der de Boor–Algorithmus an einem m–fachen Knoten t_j (nur "auseinandergezogen" gezeichet), für $m = 3$. Die Zwischenpunkte in jedem Schritt sind die Kontrollpunkte und am Schluß bleibt nur $\boldsymbol{d}_{j-m}$ übrig.

$\boldsymbol{d}_{j-m}$. An einem $m+1$–fachen Knoten werden also sogar $\boldsymbol{d}_{j-m}$ und $\boldsymbol{d}_{j-m+1}$ interpoliert, die Kurve wird dort also normalerweise nicht mehr stetig sein.

4. Spezialisieren wir noch ein bißchen: Ist obendrein noch $m = n$, haben wir also

$$t_0 = \cdots = t_n < t_{n+1} = \cdots = t_{2n+1},$$

dann ist der Algorithmus von de Boor nichts anderes als der Algorithmus von de Casteljau und es ist

$$N_{n,T}\boldsymbol{d} = B_n\boldsymbol{d}.$$

5. Für alle $x \in (t_r, t_{r+1})$[90] werden *dieselben* Bezugsintervalle I_k^{m-j+1} verwendet und damit

[86] Oder der Ordnung $m+1$! Es gibt hier (mindestens) zwei konkurrierende Terminologien. Daher ist bei der Verwendung von Splineliteratur oder von Spline–Toolboxen (z.B. in `Matlab`) *erhöhte* Vorsicht angesagt.

[87] Genauer: auf den beiden Intervallen $[t_m, t_{m+1}]$ und $[t_n, t_{n+1}]$

[88] Wegen $r = m$

[89] Schließlich ist $t_j = \cdots = t_m$, $j = 1, \ldots, m$.

[90] Achtung: Ist $t_r = t_{r+1}$, dann ist $(t_r, t_{r+1}) = \emptyset$!

ist $N_{m,T}\boldsymbol{d}|_{(t_r,t_{r+1})}$ ein Produkt von m affinen Funktionen, also ein Polynom vom Grad $\leq m$. Anders gesagt:

Splinekurven sind stückweise Polynome!

6. Da
$$N_{m,T}\left(\alpha\boldsymbol{d}+\beta\boldsymbol{d}'\right)=\alpha\ N_{m,T}\boldsymbol{d}+\beta\ N_{m,T}\boldsymbol{d}'$$
können wir $\boldsymbol{d}$ als
$$\boldsymbol{d}=\sum_{j=0}^{n}\boldsymbol{d}_j\ \delta_j,\qquad \delta_j=\ (0,\ldots,0,\ \underset{\substack{\uparrow\\ j}}{1},\ 0,\ldots,0).\ ,$$
und damit die Splinekurve als Linearkombination
$$N_{m,T}\boldsymbol{d}(x)=\sum_{j=0}^{n}\boldsymbol{d}_j\ N_j^m\left(x|T\right). \tag{8.5}$$
schreiben.

7. Wie erhalten wir also die sogenannten "B–Splines"? Ganz einfach: Wir lassen den Algorithmus von de Boor auf die *skalaren* Kontrollpunkte δ_j los.

Definition 8.5 *Die Funktion $N_j^m\left(\cdot|T\right)$, aus (8.5) heißt j–ter* B–Spline *der Ordnung m bezüglich der Knotenfolge T.*

8.2 Splines und Interpolation

Im Zusammenhang mit dieser Vorlesung wollen wir uns im wesentlichen mit der Frage befassen, ob und wann es möglich ist, mit Splines zu interpolieren. Das wesentliche Resultat, mit dessen Beweis wir uns den Rest dieses Kapitels beschäftigen werden, ist wie folgt.

Satz 8.6 *Es sei $T=T_{m,n}$ eine Knotenfolge und es seien $x_0<x_1<\cdots<x_n\in\mathbb{R}$.*

1. *(* Schoenberg–Whitney *) Das Interpolationsproblem*
$$N_{m,T}\boldsymbol{d}\left(x_j\right)=\boldsymbol{f}_j,\qquad j=0,\ldots,n, \tag{8.6}$$
ist genau dann für alle $\boldsymbol{f}\in\mathbb{R}^{d\times n+1}$ eindeutig lösbar, wenn
$$t_j<x_j<t_{j+m+1},\qquad j=0,\ldots,n. \tag{8.7}$$

2. *Ist (8.7) erfüllt, dann ist die* Kollokationsmatrix
$$\left[N_k^m\left(x_j\mid T\right):j,k=0,\ldots,n\right] \tag{8.8}$$
(m,m)–bandiert und total nichtnegativ.

Bemerkung 8.7 *1. Die Bedingung (8.7), die die Knoten und die Interpolationspunkte miteinander in Beziehung setzt, kann man auch anders schreiben, wenn man j durch $j-m-1$ ersetzt und das Ergebnis, $t_{j-m-1} < x_{j-m-1} < t_j$, mit (8.7) zu*

$$x_{j-m-1} < t_j < x_j, \qquad j = m+1, \ldots, n, \tag{8.9}$$

zusammenfasst, was gerade Bedingungen an die Lage der inneren *Knoten liefert, siehe Definition 8.1.*

2. Zusammen mit (8.7) kann $x_j \in (t_k, t_{k+m+1})$ aber nur dann gelten, wenn

$$j \le k+m \quad \textit{und} \quad j+m \le k \qquad \Rightarrow \qquad k-m \le j \le k+m$$

ist. Da, wie wir in Lemma 8.14 herausfinden werden, der B–Spline $N_k^m(\cdot \mid T)$ aber nur im offenen Intervall (t_k, t_{k+m+1}) von Null verschieden[91] *ist, liefert diese Beobachtung auch bereits die (m,m)–Bandiertheit der Kollokationsmatrix (8.8).*

Bevor wir uns daran machen, uns die Splines genauer anzusehen, erst einmal ein einfaches Beispiel, das Satz 8.6 ein bißchen illustriert.

Beispiel 8.8 *Wir setzen $m = 1$ und betrachten eine Knotenfolge $T = T_{n,1} = t_0 < \cdots < t_{n+2}$*[92]*.*

1. Im Algorithmus von de Boor müssen wir einen Schritt berechnen, bei dem die baryzentrischen Koordinaten von x bezüglich des Intervalls $[t_k, t_{k+1}]$ verwendet werden, um $\boldsymbol{d}_{k-1}$ und $\boldsymbol{d}_k$ zu kombinieren, $k = 1, \ldots, n$. Damit ist aber $N_{1,T}\boldsymbol{d}$ nichts anderes als die stückweise lineare Funktion mit "Bruchstellen" an $t_1, \ldots, t_n$, die die Punkte $\boldsymbol{d}_0, \ldots, \boldsymbol{d}_n$ verbindet.

2. Damit können wir $N_{1,T}\boldsymbol{d}$ als

$$N_{1,T}\boldsymbol{d} = \sum_{j=0}^{n} \boldsymbol{d}_j \; N_j^1(\cdot|T)$$

schreiben, wobei

$$N_j^1(x|T) = \begin{cases} 0 & x < t_j, \\ \frac{x-t_j}{t_{j+1}-t_j} & x \in [t_j, t_{j+1}], \\ \frac{t_{j+2}-x}{t_{j+2}-t_{j+1}} & x \in [t_{j+1}, t_{j+2}], \\ 0 & x > t_{j+2}. \end{cases}$$

[91] Und sogar strikt positiv!

[92] Es sind jetzt also *alle* Knoten, auch die Randknoten, *einfach* – aber nur um der Einfachheit willen.

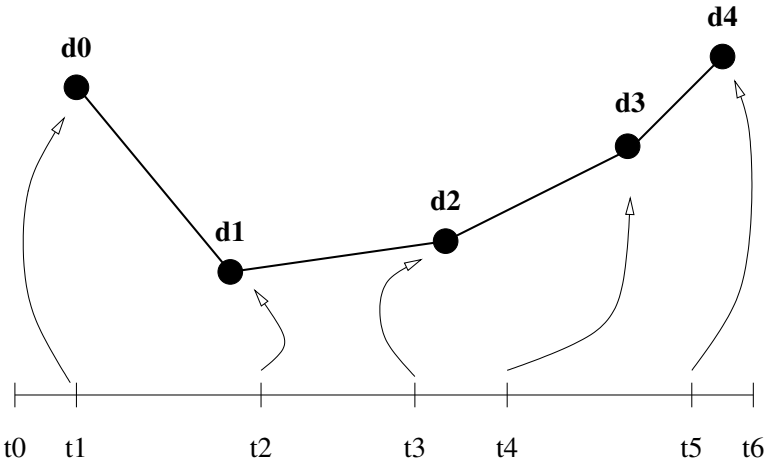


Abbildung 8.3: Die lineare Splinekurve (Ordnung $m = 1$).

3. *Was bedeutet nun die Schoenberg–Whitney–Bedingung (8.7)? Zuerst einmal muß der Interpolationspunkt x_j im* offenen *Intervall $[t_j, t_{j+2}]$, also im Träger von N_j^1, liegen. Das garantiert also schon einmal, daß aller Basisfunktionen beim Interpolieren mitmachen dürfen – übrigens, wäre das nicht der Fall, dann hätte die Kollookationsmatrix eine Nullzeile, wäre nicht invertierbar und damit das Interpolationsproblem unlösbar.*

   Umgekehrt garantiert (8.7) aber auch, daß in jedem Knotenintervall höchstens zwei *Interpolationspunkte liegen – sehr vernünftig, denn durch zwei Punkte läßt sich gerade noch eine Gerade legen. Drei oder mehr Interpolationspunkte in einem Knotenintervall ergäben im allgemeinen wieder ein unlösbares Interpolationsproblem.*

   Liegen außerdem zwei Punkte im Inneren eines Knotenintervalls, dann darf in den benachbarten Intervallen nur jeweils ein Knoten liegen; lägen nämlich in zwei benachbarten Intervallen zwei Interpolationspunkte, dann ist es einfach, die Interpolationsbedingungen an diesen Punktpaaren so festzulegen, daß sich die beiden Streckenzüge nicht *in dem dazwischenliegenden Knoten schneiden – und damit ist dann auch der Spline futsch.*

4. *Insbesondere liefert (8.7) aber auch, daß N_j^1 höchstens an x_{j-1}, x_j und x_{j+1} einen von Null verschiedenen Wert hat (und wenn verschieden, dann > 0). Also hat die Kollokationsmatrix die Form*

$$\begin{bmatrix} * & * & & & \\ * & * & * & & \\ & \ddots & \ddots & \ddots & \\ & & * & * & * \\ & & & * & * \end{bmatrix},$$

   ist also $(1,1)$–bandiert oder tridiagonal. *Die totale Positivität ist allerdings nicht so offensichtlich.*

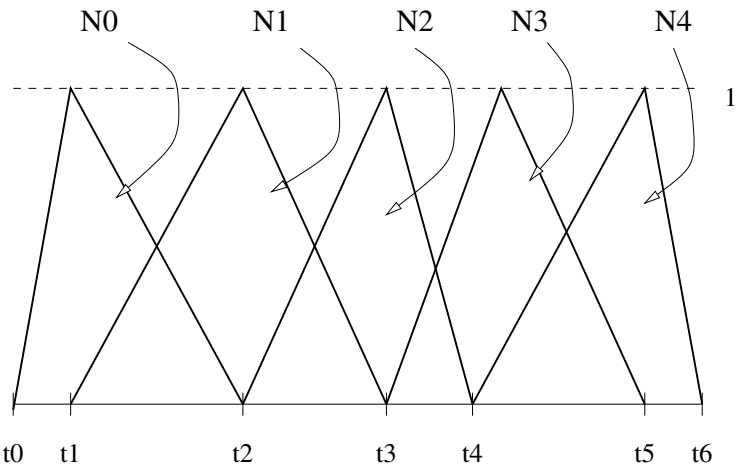


Abbildung 8.4: Die zugehörigen vier B–Splines zur Splinekurve aus Abb 8.3.

5. *Es gibt noch einen besonders einfachen Fall: $x_j = t_{j+1}$. Dann ist die Kollokationsmatrix sogar die $(0,0)$–bandierte Einheitsmatrix und $\boldsymbol{d}_j = \boldsymbol{f}_j$, aber so einfach ist's halt nicht immer.*

Als Folgerung aus dem Satz von Schoenberg–Whitney können wir einen besonderen Splineinterpolanten herleiten.

Satz 8.9 *Sei $m = 2r+1 \in \mathbb{N}$ und $T = T_{m,n}$ eine Knotenfolge mit* einfachen *Knoten. Dann gibt es zu vorgegebenen Werten $\boldsymbol{f}_j$, $j = 0, \ldots, n-m+1$, einen Spline $N_{m,T}\ \boldsymbol{d}$, so daß*[93]

$$N_{m,T}\ \boldsymbol{d}\,(t_{m+j}) = \boldsymbol{f}_j, \qquad j = 0, \ldots, n-m+1, \tag{8.10}$$

$$N^{(k)}_{m,T}\ \boldsymbol{d}\,(t_\ell) = 0, \qquad k = r+1, \ldots, 2r,\ \ell = m, n+1. \tag{8.11}$$

Ist $n \geq m + r$, dann ist dieser interpolierende Spline eindeutig.

Definition 8.10 *Der*[94] *Spline, der die Bedingungen (8.10) und (8.11) erfüllt, heißt* "natürlicher Splineinterpolant" *an den Knoten $t_m, \ldots, t_{n+1}$.*

Beweis: Nach dem Satz von Schoenberg–Whitney, Satz 8.6, ist bei einfachen Knoten das Splineinterpolationsproblem für

$$x_j = t_{j+r+1} \in (t_j, t_{j+2r+1}), \qquad j = 0, \ldots, n,$$

eindeutig lösbar. Dabei verwenden wir, zusätzlich zu den $n+2-m = n+1-2r$ Bedingungen aus (8.10) noch Interpolationsbedingungen an den "letzten" r linken und den "ersten" r rechten

[93] Der Spline interpoliert als an den inneren Knoten und den innersten Randknoten.

[94] Für $n \geq m + r$ ist es "der", sonst nur "ein" ....

Randknoten; insgesamt sind ja auf jeder Seite $m+1 = 2r+2 > r$ Randknoten. Seien nun $\boldsymbol{s}_{-r}, \ldots, \boldsymbol{s}_r$ die Lösungen des Interpolationsproblems, das man erhält, wenn man zusätzlich zu (8.10) noch

$$\boldsymbol{s}_j(x_k) = \begin{cases} 1 & j < 0 \text{ und } k = r - j, \\ 1 & j > 0 \text{ und } k = n + j - r, \\ 0 & \text{sonst}, \end{cases} \qquad k = 0, \ldots, r-1, n+1, \ldots, n+r,$$

fordert. Alle diese Splines interpolieren an den Punkten $t_m, \ldots, t_{n+1}$, aber haben ein unterschiedliches Verhalten an den $2r$ zusätzlichen Punkten $t_{r+1}, \ldots, t_{m-1}$ und $t_{n+2}, \ldots, t_{n+r+1}$: $\boldsymbol{s}_0$ verschwindet dort, während $\boldsymbol{s}_{-1}, \ldots, \boldsymbol{s}_{-r}$ jeweils an einem der linken "Zusatzknoten", $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_r$ an einem der rechten "Zusatzknoten" den Wert 1 annehmen, siehe Abb. 8.5.
Diese $m = 2r+1$ Splines sind $\neq 0$ und linear unabhänging. Betrachten wir nun das lineare Gleichungssystem für $a_{-r}, \ldots, a_r$, das gegeben ist durch

$$\sum_{j=-r}^{r} a_j \, \boldsymbol{s}_j^{(k)}(t_\ell) = 0, \qquad k = r+1, \ldots, 2r, \; \ell = m, n+1,$$

dann sind dies $2r$ Gleichungen in den $2r+1$ Unbekannten $a_{-r}, \ldots, a_r$, und es gibt immer (mindestens) eine nichttriviale Lösung $a_{-r}^*, \ldots, a_r^*$, die entweder $a_{-r}^* + \cdots + a_r^* = 0$ erfüllt, oder die man so normieren kann, daß $a_{-r}^* + \cdots + a_r^* = 1$. Auf alle Fälle setzen wir

$$\boldsymbol{s} = \sum_{j=-r}^{r} a_j^* \, \boldsymbol{s}_j$$

und da für $j = 0, \ldots, n-m+1$

$$\boldsymbol{s}(t_{m+j}) = \sum_{k=-r}^{r} a_k^* \underbrace{\boldsymbol{s}_k(t_{j+m})}_{=\boldsymbol{f}_j} = \boldsymbol{f}_j \sum_{k=-r}^{r} a_k^*,$$

ist $\boldsymbol{s}$ entweder eine Lösung von (8.10) oder ein Lösung des homogenen Problems[95] zu (8.10), je nachdem, ob $\sum a_j^*$ den Wert 1 oder 0 hat, aber erfüllt immer auch zusätzlich (8.11). Wir haben also entweder unseren gewünschten Spline gefunden, und die Eindeutigkeit folgt aus dem nächsten Resultat, das auch die Namensgebung rechtfertigt, oder aber eine nichttriviale Lösung des homogenen Problems. Nach Satz 8.11 wäre dann aber $\boldsymbol{s}^{(r+1)} = 0$, also $\boldsymbol{s}$ ein Polynom vom Grad r – ein Trivialfall, der nur eintreten kann, wenn wir zu wenige innere Knoten haben, also wieder, wenn $n < m + r$ ist. In diesem Fall existiert aber unser interpolierender natürlicher Spline trivialerweise: er ist der polynomiale Interpolant vom Grad $n - m$! □

Für $k \in \mathbb{N}_0$ und $I \subset \mathbb{R}$ definieren wir die "Energienormen"[96]

$$|f|_{k,I} = \left(\int_I \left| f^{(k)}(x) \right|^2 \, dx \right)^{1/2}, \qquad f \in C^{(k)}(I).$$

[95] Also mit rechter Seite 0.
[96] Die allerdings nur Halbnormen sind.

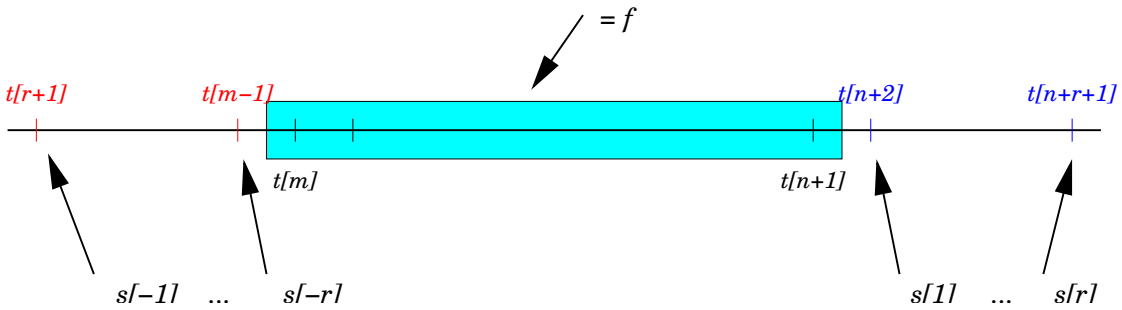


Abbildung 8.5: Das erweiterte Interpolationsproblem aus dem Beweis von Satz 8.9. Auf dem unterlegten Bereich interpolieren alle Splines die vorgegebenen Werte, an den "Zusatzpunkten" hat genau einer von ihnen den Wert 1, alle andere den Wert 0; nur s_0 verschwindet an *allen* äußeren Punkten.

Im Spezialfall $k = 2$, der, wie wir gleich sehen werden, zu $m = 3$ gehört, ist dies das Integral über das Quadrat der zweiten Ableitung, eine *Näherung* für die Biegeenergie. Daher übrigens auch der Name "Spline"[97]: Ein Spline[98] ist ein Kurvenlineal, bei dem ein mehr oder weniger beweglicher Streifen mit Hilfe von Gewichten durch gewisse Punkte gezwungen wird, siehe Abb. 8.6. Da der Streifen sich so legt, daß die Biegeenergie[99] minimiert wird, bezeichnet

Abbildung 8.6: Ein "richtiger" Spline, also das Kurvenlineal. Die moderne Form besteht aus einem nicht allzu flexiblen Plastikstreifen und (richtig schweren) Gewichten, die die Interpolationspunkte fixieren. Vielen Dank an Herrn Dr. Hollenhorst vom Hochschulrechenzentrum der JLU Gießen, der diesen Spline freundlicherweise zur Verfügung stellte.

[97] Wenn man es genau nimmt, ist also ein Spline gar kein Spline.
[98] Im Schiffsbau als *Straklatte* bezeichnet.

man den *kubischen*[100] Spline mit den den Randbedingungen aus Satz 8.11[101] als *"natürlichen Spline"*, weil er das "natürliche" Objekt "Spline" simuliert. Dabei ensteht die "Natürlichkeit" aus den Randbedingungen (8.11) für $m = 3$, also $r = 1$, also aus

$$N''_{m,T}\,\boldsymbol{d}\,(t_m) = N''_{m,T}\,\boldsymbol{d}\,(t_{n+1}) = 0 \tag{8.12}$$

am ersten und letzten Interpolationspunkt, siehe Abb. 8.7. Hierbei kann man (8.12) so inter-

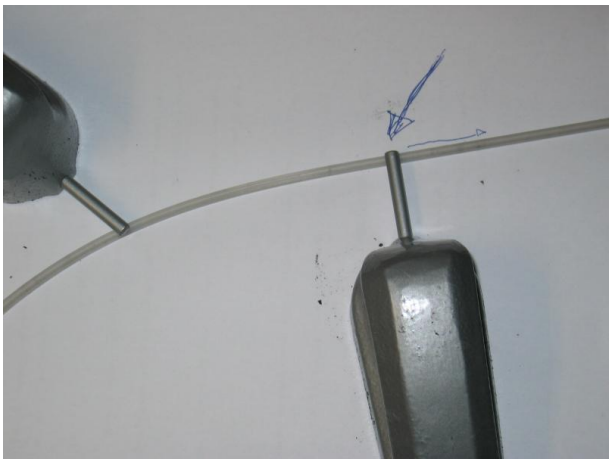


Abbildung 8.7: Die natürliche Randbedingung des kubischen Splines: Rechts vom letzten Knoten und damit Interpolationspunkt werden auf das Lineal keine Biegekräfte mehr ausgeübt und es nimmt den energieminimalen linearen Verlauf an.

pretieren, daß links von t_m und rechts von t_{n+1} die Kurve $N_{m,T}$ *linear* fortgesetzt werden soll, was auch genau das ist, was ein elastisches Material wie der Plastikstreifen tun wird, wenn keine weiteren Anstrengungen unternommen werden, ihn zu biegen.

Satz 8.11 (Minimalitätseigenschaften) *Es sei $m = 2r + 1 \in \mathbb{N}$ und $T = T_{m,n}$ eine einfache Knotenfolge. Für $f \in C^{r+1}(I)$ sei $S_f = N_{m,T}\,\boldsymbol{d}$ ein Spline, der (8.10) und (8.11) erfüllt. Dann ist*

$$|S_f|_{r+1,I} \le |f|_{r+1,I} \tag{8.13}$$

Beweis: Wir beginnen mit

$$|f - S_f|^2_{r+1,I} = \int_I \left(f^{(r+1)}(x) - (S_f)^{(r+1)}(x)\right)^2 dx$$

[99] Und die entspricht dem Quadratintegral über die zweite Ableitung, solange die Krümmung im Vergleich zur Ableitung klein ist.

[100] *Zweite* Ableitung!

[101] Die gerade für die Minimalität sorgen werden.

$$
\begin{aligned}
&= \int_I \left(f^{(r+1)}(x)\right)^2 - 2f^{(r+1)}(x)(S_f)^{(r+1)}(x) + \left((S_f)^{(r+1)}(x)\right)^2 dx \\
&= |f|^2_{r+1,I} - 2\int_I \left(f^{(r+1)}(x) - (S_f)^{(r+1)}(x)\right)(S_f)^{(r+1)}(x)\,dx - |S_f|^2_{r+1,I}\,. \qquad (8.14)
\end{aligned}
$$

Für $j = m, \ldots, n$ verwendet man partielle Integration, um zu zeigen, daß

$$
\begin{aligned}
&\int_{t_j}^{t_{j+1}} \left(f^{(r+1)}(x) - S_f^{(r+1)}(x)\right) S_f^{(r+1)}(x)\,dx \\
&= \left(f^{(r)}(x) - S_f^{(r)}(x)\right) S_f^{(r+1)}(x)\Big|_{t_j}^{t_{j+1}} - \int_{t_j}^{t_{j+1}} \left(f^{(r)}(x) - S_f^{(r)}(x)\right) S_f^{(r+2)}(x)\,dx \\
&= \sum_{l=0}^{k}(-1)^{r-l}\left(f^{(r-l)}(x) - S_f^{(r-l)}(x)\right) S_f^{(r+l+1)}(x)\Big|_{t_j}^{t_{j+1}} \\
&\qquad +(-1)^{k+1}\int_{t_j}^{t_{j+1}} \left(f^{(r-k)}(x) - S_f^{(r-k)}(x)\right) \underbrace{S_f^{(r+k+2)}(x)}_{=0 \text{ für } k=r}\,dx, \qquad k = 1, \ldots, r \\
&= \sum_{l=0}^{r}(-1)^{r-l}\left(f^{(r-l)}(x) - S_f^{(r-l)}(x)\right) S_f^{(r+l+1)}(x)\Big|_{t_j}^{t_{j+1}},
\end{aligned}
$$

Summation über $j = m, \ldots, n$ liefert dann, daß[102]

$$
\begin{aligned}
&\int_I \left(f^{(r+1)}(x) - S_f^{(r+1)}(x)\right) S_f^{(r+1)}(x)\,dx \\
&\qquad = \sum_{l=0}^{r}(-1)^{r-l}\left(f^{(r-l)}(x) - S_f^{(r-l)}(x)\right) S_f^{(r+l+1)}(x)\Big|_{t_m}^{t_{n+1}} = 0.
\end{aligned}
$$

Beachte: Der Term für $l = r$ verschwindet, weil S_f die Funktion f an den Stellen t_m, t_{n+1} interpoliert, die Terme für $l = 0, \ldots, r-1$ hingegen wegen (8.11). Einsetzen in (8.14) liefert schließlich

$$|S_f|^2_{r+1,I} = |f|^2_{r+1,I} - |f - S_f|^2_{r+1,I} \le |f|^2_{r+1,I}\,.$$

Hierbei gilt Gleichheit genau dann, wenn $f - S_f \in \Pi_r$.
Das zeigt auch die Eindeutigkeit des natürlichen Splineinterpolanten wenn $n \ge m + r$: Ist f irgendeine andere Minimallösung des Interpolationsproblems, dann ist $f - S_f \in \Pi_r$, muß aber an mindestens $r + 1$ Stellen (den Interpolationspunkten) verschwinden. Also ist $f = S_f$. □

Übung 8.1 Was sind die kleinsten Zahlen p, q, so daß die Kollokationsmatrix zum natürliche Splineinterpolant der Ordnung $m = 2r + 1$ eine (p, q)–bandierte Matrix ist. ♢

Bei de Boor findet man zu diesem Thema die folgende Aussage:

[102] Die Werte an den inneren Knoten tauchen stets zweimal, aber mit unterschiedlichen Vorzeichen auf.

Die Extremaleigenschaft des interpolierenden Splines wird häufig für die große praktische Nützlichkeit der Splines verantwortlich gemacht. Dies ist jedoch glatter "Volksbetrug" . . ..

8.3 Eigenschaften der B–Splines

Bevor wir uns mit B–Splines befassen, sehen wir uns in Abb. 8.8 erst einmal ein paar davon an.

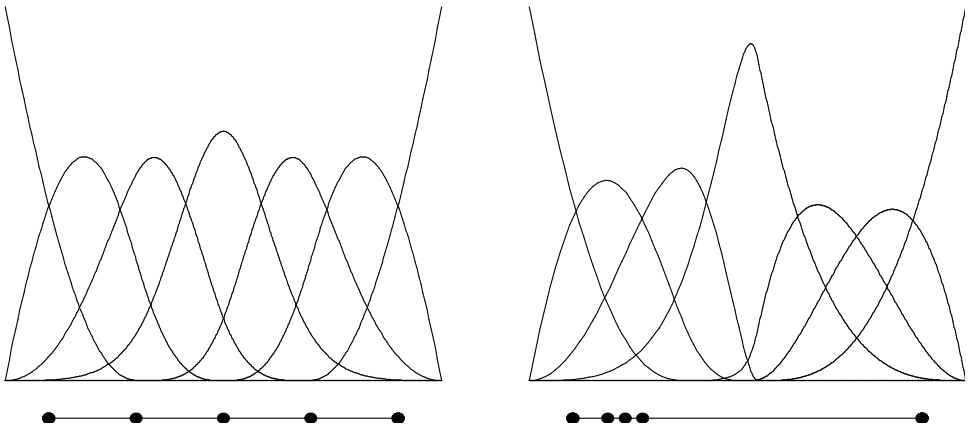


Abbildung 8.8: Eine kleine Kollektion kubischer B–Splines ($m = 3$) mit vierfachen Randknoten. In linken Bild sind die inneren Knoten gleichverteilt, im rechten Bild in der linken Hälfte konzentriert.

Das erste wichtige Resultat ist eine Rekursionsformel für B–Splines, mit deren Hilfe man B–Splines der Ordnung m aus B–Splines der Ordnung $m-1$ berechnen kann. Diese Formel geht auf Carl de Boor zurück.

Satz 8.12 (Rekursionsformel für B–Splines) *Sei $T = T_{m,n}$ eine Knotenfolge.*

1. *Die B–Splines der Ordnung 0 haben die Form*

$$N_j^0(x|T) = \chi_{[t_j,t_{j+1})}(x) = \begin{cases} 1 & x \in [t_j, t_{j+1}), \\ 0 & x \notin [t_j, t_{j+1}), \end{cases} \qquad j = 0, \ldots, n+m. \tag{8.15}$$

2. *Für $\ell \geq 1$ gilt*

$$N_j^\ell(x|T) = u_1\left(x|I_j^\ell\right) N_j^{\ell-1}(x|T) + u_0\left(x|I_{j+1}^\ell\right) N_{j+1}^{\ell-1}(x|T), \qquad j = 0, \ldots, n+m-\ell. \tag{8.16}$$

Bemerkung 8.13

1. *In den Ausdrücken $N_j^\ell(\cdot|T)$ sollte man $T = T_{m,n}$ als $T = T_{\ell,n+m-\ell}$ auffassen. Und damit gibt es eben auch nicht n sondern $n+m-\ell$ B–Splines der Ordnung ℓ zu einer vorgegebenen Knotenfolge $T = T_{m,n}$.*

2. *Explizit geschrieben lautet (8.16)*

$$N_j^\ell(x|T) = \frac{x - t_j}{t_{j+\ell} - t_j} N_j^{\ell-1}(x|T) + \frac{t_{j+\ell+1} - x}{t_{j+\ell+1} - t_{j+1}} N_{j+1}^{\ell-1}(x|T), \tag{8.17}$$

   $j = 0, \ldots, n+m-\ell$.

3. *Die Formel (8.17) ist für $\ell+1$–fache Knoten nicht definiert – kein Wunder, da dann auch der Spline $N_j^{\ell-1}$ oder $N_{j+1}^{\ell-1}$ nicht mehr vernünftig definiert ist*[103]*. In diesem Fall läßt man den entsprechenden Term in (8.17) einfach wegfallen.*

Lemma 8.14 *Die B–Splines sind nichtnegative Funktionen mit kompaktem Träger. Genauer:*

$$N_j^m(x|T) > 0, \quad x \in (t_j, t_{j+m+1}), \qquad \textit{und} \qquad N_j^m(x|T) = 0, \quad x \notin [t_j, t_{j+m+1}]. \tag{8.18}$$

Beweis: Wir erinnern uns, daß zur Bestimmung des Wertes an der Stelle $x \in [t_r, t_{r+1})$ gerade die Kontrollpunkte $d_{r-m}, \ldots, d_r$ verwendet werden. Um $N_j^m(x|T) \neq 0$ zu erhalten, muß also $j \in \{r-m, \ldots, r\}$ oder eben $r \in \{j, \ldots, j+m\}$ gelten. Damit verschwindet der B–Spline $N_j^m(x|T)$ außerhalb von $[t_j, t_{j+m+1}]$. Gilt andererseits $x \in (t_r, t_{r+1})$ und ist $r \in \{j, \ldots, j+m\}$, dann sind wegen

$$(t_r, t_{r+1}) \subseteq \begin{cases} (t_j, t_{j+m-k+1}), & j = r-m+k, \ldots, r, \\ I_r^{m-k+1}, \end{cases} \qquad , k = 1, \ldots, m,$$

alle baryzentrischen Koordinaten in (8.3) strikt positiv und damit ist auch

$$d_j^1(x) = u_0\left(x|I_j^m\right) \underbrace{d_{j-1}^0(x)}_{=d_{j-1}=0} + u_1\left(x|I_j^m\right) \underbrace{d_j^0(x)}_{=d_j=1} = u_1\left(x|I_j^m\right) > 0$$

sowie

$$d_{j+1}^1(x) = u_0\left(x|I_{j+1}^m\right) \underbrace{d_j^0(x)}_{=d_j=1} + u_1\left(x|I_{j+1}^m\right) \underbrace{d_j^0(x)}_{=d_{j+1}=0} = u_0\left(x|I_{j+1}^m\right) > 0.$$

Mit demselben Argument[104] ergibt sich dann, daß

$$d_k^\ell(x) > 0, \qquad k = j, \ldots, j+\ell$$

Also ist auch $N_j^m(x|T) = d_r^m(x) > 0$, da $r \in \{j, \ldots, j+m\}$. □

[103] Genauer: Der Spline wäre, wie wir gleich sehen werden, nur auf der leeren Menge von Null verschieden.

[104] Man ersetzt $d_j^1(x)$ durch $d_j^{\ell+1}(x)$

Lemma 8.15 *Die B–Splines bilden eine nichtnegative Teilung der Eins, d.h.*

$$\sum_{j=0}^{n} N_j^m\left(\cdot|T\right) \equiv 1. \tag{8.19}$$

Beweis: Ist $d_0 = \cdots = d_n = 1$, also $d_k^0(x) = 1$, $k = 0, \ldots, n$, dann ist, nach (8.4) und Induktion über j,

$$\begin{aligned} d_k^j(x) &= u_0\left(x|I_k^{m-j+1}\right) d_{k-1}^{j-1}(x) + u_1\left(x|I_k^{m-j+1}\right) d_k^{j-1}(x) \\ &= u_0\left(x|I_k^{m-j+1}\right) + u_1\left(x|I_k^{m-j+1}\right) = 1, \qquad\qquad k = j, \ldots, m. \end{aligned}$$

□

Beweis von Satz 8.12: Nach Lemma 8.14 haben die B–Splines $N_j^0\left(\cdot|T\right)$, $j = 0, \ldots, n+m$, als Träger das Intervall $[t_j, t_{j+1})$, insbesondere sind also die Träger der verschiedenen B–Splines disjunkt und Lemma 8.15 liefert (8.15).
Um $N_j^m\left(\cdot|T\right)$ mit Hilfe der B–Splines der Ordnung $m-1$ ausdrücken zu können, benutzen wir die Zwischenpunkte $d_j^1(x)$, $j = 1, \ldots, n$, aus (8.4) und schreiben

$$N_{m,T}\boldsymbol{d}(x) = \sum_{j=0}^{n} d_j N_j^m\left(x|T\right) = \sum_{j=1}^{n} d_j^1(x)\; N_j^{m-1}\left(x|T\right). \tag{8.20}$$

Ist insbesondere $d_j = \delta_{jk}$, dann ist, nach (8.3)

$$d_j^1(x) = u_0\left(x|I_j^m\right) d_{j-1} + u_1\left(x|I_j^m\right) d_j = \begin{cases} u_1\left(x|I_k^m\right) & j = k, \\ u_0\left(x|I_{k+1}^m\right) & j = k+1, \\ 0 & \text{sonst.} \end{cases} \tag{8.21}$$

Setzt man (8.21) in (8.20) ein, dann ergibt sich

$$N_j^m\left(x|T\right) = u_1\left(x|I_j^m\right) N_j^{m-1}\left(x|T\right) + u_0\left(x|I_{j+1}^m\right) N_{j+1}^{m-1}\left(x|T\right). \tag{8.22}$$

□

8.4 Der Splineraum

Das "B" bei den B–Splines kommt natürlich nicht von ungefähr: die B–Splines bilden eine Basis eines bestimmten Raums von stückweise polynomialen Funktionen.

Definition 8.16 *Zu $m \in \mathbb{N}$ und einer Knotenfolge $T = T_{m,n}$ ist der* Splineraum $\mathbb{S}_m(T)$ *definiert als Menge aller*

1. *stückweisen Polynome*[105]

$$f \in \mathbb{S}_m(T) \qquad \Longrightarrow \qquad f|_{(t_j, t_{j+1})} \in \Pi_m, \quad j = m, \ldots, n, \tag{8.23}$$

[105] Nicht vergessen: Eingschränkt auf die leere Menge hat jede Funktion jede Eigenschaft, deswegen – vorsichtshalber – die offenen Intervalle.

2. *die an einem μ–fachen Knoten t_j, $t_{j-1} < t_j = \cdots = t_{j+\mu-1} < t_{j+\mu}$, differenzierbar von der Ordnung $m - \mu$ sind:*

$$f \in \mathbb{S}_m(T) \qquad \Longrightarrow \qquad f \in C^{m-\mu}\left(t_{j-1}, t_{j+\mu}\right). \tag{8.24}$$

Offensichtlich ist $\mathbb{S}_m(T)$ ein Vektorraum, das heißt,

$$f, g \in \mathbb{S}_m(T) \qquad \Longrightarrow \qquad \alpha f + \beta g \in \mathbb{S}_m(T), \quad \alpha, \beta \in \mathbb{R}.$$

Satz 8.17 (Curry–Schoenberg) *Für jedes $m \in \mathbb{N}$ und jede Knotenfolge $T = T_{m,n}$ sind die B–Splines $N_j^m\left(\cdot|T\right)$ eine Basis von $\mathbb{S}_m(T)$.*

Für diesen Satz ist mehr zu beweisen, als man zuerst denken mag: zwar folgt (8.23) sofort aus dem Algorithmus von de Boor, aber wir müssen trotzdem noch zeigen, daß die B–Splines *linear unabhängig* sind, die gewünschten Differenzierbarkeitseigenschaften haben und daß die Dimension von $\mathbb{S}_m(T)$ tatsächlich $n + 1$ ist. Es gibt einen sehr schönen Beweis von Satz 8.17, bei dem man die Kontrollpunkte zu einer stückweise polynomialen Kurve direkt angeben kann, aber dieser beruht auf dem sogenannten "Blossoming Principle" und ist eher etwas für eine Spezialvorlesung[106]. Hier wollen wir den direkten Weg gehen. Dafür müssen wir aber eine zusätzliche Annahme machen, indem wir fordern, daß

$$t_m < t_{m+1} \qquad \text{und} \qquad t_n < t_{n+1}, \tag{8.25}$$

daß also kein Randknoten als innerer Knoten auftritt[107]. Dies ist insbesondere für $(m+1)$–fache Randknoten der Fall.

Eine sehr hilfreich Aussage, die uns mit ein bißchen Rechenarbeit konfrontieren wird, ist eine Formel für die Ableitung eines B–Splines.

Lemma 8.18 *Sei $m \in \mathbb{N}$ und $T = T_{m,n}$ eine Knotenfolge. Dann ist, für $x \in \mathbb{R} \setminus T$,*

$$\frac{d}{dx} N_j^m\left(x|T\right) = \frac{m}{t_{j+m} - t_j} N_j^{m-1}\left(x|T\right) - \frac{m}{t_{j+m+1} - t_{j+1}} N_{j+1}^{m-1}\left(x|T\right), \qquad j = 0, \ldots, n. \tag{8.26}$$

Ist t_j oder t_{j+1} ein Knoten der Vielfachheit $m + 1$, dann macht (8.26) zuerst einmal keinen Sinn. Allerdings ist in diesem Fall auch das Trägerintervall des betreffenden B–Splines N_j^{m-1} oder N_{j+1}^{m-1} auch eine einpunktige Teilmenge von T, was die Division durch 0 wieder kompensiert.

Beweis: Induktion über m, wobei der Fall $m = 1$ sich sehr einfach "von Hand" überprüfen lässt. Da N_j^m auf jeder offenen und konvexen Teilmenge $U \subset R \setminus T$[108] ein Polynom ist, können

[106] leider ...

[107] Klingt irgendwo plausibel.

[108] Wir bleiben also immmer in einem der "Stückchen", in die T die relle Achse zerlegt.

wir immer und überall nach Herzenslust differenzieren. Unter Verwendung von (8.17) ergibt sich dann

$$\begin{aligned}
&\left(\frac{d}{dx}N_j^m\left(\cdot|T\right)\right)(x)\\
&= \left(\frac{d}{dx}\left(\frac{\cdot - t_j}{t_{j+m}-t_j}N_j^{m-1}\left(\cdot|T\right) + \frac{t_{j+m+1}-\cdot}{t_{j+m+1}-t_{j+1}}N_{j+1}^{m-1}\left(\cdot|T\right)\right)\right)(x)\\
&= \frac{1}{t_{j+m}-t_j}N_k^{m-1}\left(\cdot|T\right) - \frac{1}{t_{j+m+1}-t_{j+1}}N_{j+1}^{m-1}\left(\cdot|T\right)\\
&\quad + \frac{x-t_j}{t_{j+m}-t_j}\frac{d}{dx}\left(N_j^{m-1}\left(\cdot|T\right)\right)(x) + \frac{t_{j+m+1}-x}{t_{j+m+1}-t_{j+1}}\frac{d}{dx}\left(N_{j+1}^{m-1}\left(\cdot|T\right)\right)(x). \qquad (8.27)
\end{aligned}$$

Unter Verwendung der Induktionshypothese und mit (8.17) ergeben sich dann

$$\begin{aligned}
&\frac{x-t_j}{t_{j+m}-t_j}\frac{d}{dx}\left(N_j^{m-1}\left(\cdot|T\right)\right)(x)\\
&= \frac{x-t_j}{t_{j+m}-t_j}\left(\frac{m-1}{t_{j+m-1}-t_j}N_j^{m-2}\left(x|T\right) - \frac{m-1}{t_{j+m}-t_{j+1}}N_{j+1}^{m-2}\left(x|T\right)\right)\\
&= \frac{m-1}{t_{j+m}-t_j}\underbrace{\left(\frac{x-t_j}{t_{j+m-1}-t_j}N_j^{m-2}\left(x|T\right) + \frac{t_{j+m}-x}{t_{j+m}-t_{j+1}}N_{j+1}^{m-2}\left(x|T\right)\right)}_{=N_j^{m-1}(x|T)}\\
&\quad - \frac{m-1}{t_{j+m}-t_j}\underbrace{\left(\frac{t_{j+m}-x}{t_{j+m}-t_{j+1}} + \frac{x-t_j}{t_{j+m}-t_{j+1}}\right)}_{=(t_{j+m}-t_j)/(t_{j+m}-t_{j+1})}N_{j+1}^{m-2}\left(x|T\right)\\
&= \frac{m-1}{t_{j+m}-t_j}N_j^{m-1}\left(x|T\right) - \frac{m-1}{t_{j+m}-t_{j+1}}N_{j+1}^{m-2}\left(x|T\right) \qquad (8.28)
\end{aligned}$$

und

$$\begin{aligned}
&\frac{t_{j+m+1}-x}{t_{j+m+1}-t_{j+1}}\frac{d}{dx}\left(N_{j+1}^{m-1}\left(\cdot|T\right)\right)(x)\\
&= \frac{t_{j+m+1}-x}{t_{j+m+1}-t_{j+1}}\left(\frac{m-1}{t_{j+m}-t_{j+1}}N_{j+1}^{m-2}\left(x|T\right) - \frac{m-1}{t_{j+m+1}-t_{j+2}}N_{j+2}^{m-2}\left(x|T\right)\right)\\
&= \frac{m-1}{t_{j+m+1}-t_{j+1}}\underbrace{\left(\frac{x-t_{j+1}}{t_{j+m}-t_{j+1}} + \frac{t_{j+m+1}-x}{t_{j+m}-t_{j+1}}\right)}_{=(t_{j+m+1}-t_{j+1})/(t_{j+m}-t_{j+1})}N_{j+1}^{m-2}\left(x|T\right)\\
&\quad - \frac{m-1}{t_{j+m+1}-t_{j+1}}\underbrace{\left(\frac{x-t_{j+1}}{t_{j+m}-t_{j+1}}N_{j+1}^{m-2}\left(x|T\right) + \frac{t_{j+m+1}-x}{t_{j+m+1}-t_{j+2}}N_{j+2}^{m-2}\left(x|T\right)\right)}_{=N_{j+1}^{m-1}(x|T)}\\
&= \frac{m-1}{t_{j+m}-t_{j+1}}N_{j+1}^{m-2}\left(x|T\right) - \frac{m-1}{t_{j+m+1}-t_{j+1}}N_{j+1}^{m-1}\left(x|T\right). \qquad (8.29)
\end{aligned}$$

Einsetzen von (8.28) und (8.29) in (8.27) ergibt dann

$$\begin{aligned}
&\left(\frac{d}{dx}N_j^m\left(\cdot|T\right)\right)(x)\\
&= \frac{1}{t_{j+m}-t_j}N_j^{m-1}\left(x|T\right) - \frac{1}{t_{j+m+1}-t_{j+1}}N_{j+1}^{m-1}\left(x|T\right) + \frac{m-1}{t_{j+m}-t_j}N_j^{m-1}\left(x|T\right)\\
&\quad -\frac{m-1}{t_{j+m}-t_{j+1}}N_{j+1}^{m-2}\left(x|T\right) + \frac{m-1}{t_{j+m}-t_{j+1}}N_{j+1}^{m-2}\left(x|T\right) - \frac{m-1}{t_{j+m+1}-t_{j+1}}N_{j+1}^{m-1}\left(x|T\right)\\
&= \frac{m}{t_{j+m}-t_j}N_k^{m-1}\left(x|T\right) - \frac{m}{t_{j+m+1}-t_{j+1}}N_{j+1}^{m-1}\left(x|T\right),
\end{aligned}$$

und somit (8.26). □

Übung 8.2 Verifizieren Sie (8.26) für $m = 1$. Was bedeutet das geometrisch? ♢

Damit wissen wir auch, wie die Ableitung einer Splinekurve (bis auf die Werte an den Knoten) aussieht.

Korollar 8.19 *Es sein $m \in \mathbb{N}$ und $T = T_{m,n}$ eine Knotenfolge. Dann ist, für alle $x \in \mathbb{R} \setminus T$,*

$$\frac{d}{dx}N_{m,T}\,\boldsymbol{d}(x) = m\sum_{j=0}^{n+1}\frac{\boldsymbol{d}_j - \boldsymbol{d}_{j-1}}{t_{j+m}-t_j}N_j^{m-1}\left(x|T\right), \tag{8.30}$$

wobei $\boldsymbol{d}_{n+1} = \boldsymbol{d}_{-1} = 0$[109].

Beweis: Mit Lemma 8.18 ist

$$\begin{aligned}
\frac{d}{dx}N_{m,T}\,\boldsymbol{d}(x) &= m\sum_{j=0}^{n}\boldsymbol{d}_j\left(\frac{N_j^{m-1}\left(x|T\right)}{t_{j+m}-t_j} - \frac{N_{j+1}^{m-1}\left(x|T\right)}{t_{j+1+m}-t_{j+1}}\right)\\
&= m\sum_{j=0}^{n+1}\frac{\boldsymbol{d}_j - \boldsymbol{d}_{j-1}}{t_{j+m}-t_j}N_j^{m-1}\left(x|T\right).
\end{aligned}$$

□

Proposition 8.20 *Sei $m \in \mathbb{N}_0$ und $T = T_{m,n}$. Dann ist $N_j^m\left(\cdot|T\right) \in \mathbb{S}_m(T)$, $j = 0, \ldots, n$.*

Beweis: Induktion über m[110]. Für $m = 0$ müssen alle Knoten einfach sein (maximale Vielfachheit ist $m+1 = 1$) und die B–Splines sind stückweise konstante Funktionen und gehören damit *trivialerweise* zu $C^{-1}(\mathbb{R})$.
Für $m-1 \to m$, $m \geq 1$, folgt die stückweise Polynomialität direkt aus dem Algorithmus von de Boor, wir brauchen uns also nur noch um die Differenzierbarkeit zu kümmern. An einem

[109] Und wieder $N_j^{m-1}\left(x|T\right) / \left(t_{j+m} - t_j\right) \equiv 0$.
[110] Wie soll man Rekursionsformeln auch sonst ausnutzen?

$m+1$–fachen Knoten liefert uns ja bereits der de–Boor–Algorithmus die Unstetigkeit: Der Spline muß an dieser Stelle den Wert 1 und 0 haben. Hat hingegen ein Knoten $t \in (t_j, t_{j+m+1})$[111] die Vielfachheit $\mu \leq m$, dann können wir in einem offenen Intervall U mit $U \cap T = \{t\}$ differenzieren und die Ableitungsfunktion ist nach (8.26) wohldefiniert (auch an der Stelle t) und in U nach Induktionsvoraussetzung $(m-1-\mu)$–mal stetig differenzierbar. Also ist N_j^m in U $(m-\mu)$–mal stetig differenzierbar. □

Lemma 8.21 *Sei $m \in \mathbb{N}$ und $T_{m,n}$ eine Knotenfolge. Dann sind die B–Splines $N_j^m(\cdot|T)$ linear unabhängig.*

Beweis: Induktion über m. Der Fall $m = 0$ ist trivial, da die B–Splines alle disjunkten Träger haben. Für den Schluß $m \to m+1$ nehmen wir an, kein Knoten hätte Vielfachheit $m+1$ und es gäbe einen Knotenvektor $\boldsymbol{d} = (d_j : j = 0, \ldots, n)$, so daß

$$0 = N_{m,T}\, \boldsymbol{d} = \sum_{j=0}^{n} d_j\, N_j^m(\cdot|T)\,. \tag{8.31}$$

Ableitung beider Seiten liefert für alle $x \in \mathbb{R} \setminus T$, daß

$$0 = \frac{d}{dx} N_{m,T}\, \boldsymbol{d} = m \sum_{j=0}^{n+1} \frac{d_j - d_{j-1}}{t_{j+m} - t_j} N_j^{m-1}(\cdot|T)\,.$$

Nach Proposition 8.20 ist die rechte Seite stetig[112] und nach der Induktionshypothese muß schließlich $d_j = d_{j-1}$ sein, also

$$0 = d_0 = d_1 = d_2 = \cdots = d_n = d_{n+1} = 0.$$

□

Übung 8.3 Ergänzen Sie den Beweis von Lemma 8.21 um den Fall $(m+1)$–facher Knoten. (*Hinweis:* Jeder $(m+1)$–facher innerer Knoten zerlegt die Splinekurven in Teilkurven, und man kann alles auf den Fall $(m+1)$–facher *Randknoten* zurückführen)

Lemma 8.22 *Für $m \in \mathbb{N}_0$ und eine Knotenfolge $T = T_{m,n}$ ist*

$$\dim\, \mathbb{S}_m(T) = n+1.$$

Beweis: Auf dem, nach (8.25) nichtleeren Intervall $I_m = (t_m, t_{m+1})$ definieren wir ein beliebiges Polynom p_m. Sei nun $t_{m+1} = t_{m+2} = \cdots = t_{m+\mu} < t_{m+\mu+1}$ ein Knoten der Vielfachheit μ. Wir betrachten ein Polynom p_{m+1} auf $(t_{m+1}, t_{m+\mu+1})$ und schreiben es als

$$p_{m+1} = \sum_{j=0}^{m} \frac{a_j}{j!} (x - t_{m+1})^j\,.$$

[111] Also im Inneren des Tägers des B–Splines $N_j^m(\cdot|T)$.

[112] Wir haben keine $(m+1)$–fachen Knoten mehr.

Die Differenzierbarkeitsbedingungen sind nun

$$p_m^{(j)}(t_{m+1}) = p_{m+1}^{(j)}(t_{m+1}) = a_j, \qquad j = 0, \ldots, m-\mu.$$

Also hat der Raum aller stückweisen Polynome mit der geforderten Differenzierbarkeit auf den erstem beiden Intervallen die Dimension $m+1+\mu$ – die vollen $m+1$ Freiheitsgrade von p_m und die μ noch freien Parameter von p_{m+1}. Hat nun $t_{m+\mu+1}$ seinerseits Vielfachheit ν, so kommen weitere ν freie Parameter hinzu und so weiter. Wir haben also für die Knotenfolge

$$T_\ell = \{t_m, \ldots, t_{m+\ell+1}\}, \qquad t_m < t_{m+1}, \quad t_{m+\ell} < t_{m+\ell+1},$$

(plus geeignete Randknoten), daß $\dim \mathbb{S}_m(T_\ell) = m + \ell + 1$. Für $\ell = n - m$ folgt dann die Behauptung. □

Beweis von Satz 8.17: Der Beweis ist jetzt ein einfaches Spiel mit den schon bewiesenen Bauklötzen: Nach Proposition 8.20 ist

$$\text{span}\left\{N_j^m(\cdot|T) \;:\; j = 0, \ldots, n\right\} \subseteq \mathbb{S}_m(T),$$

aber da wegen der linearen Unabhängigkeit der B–Splines (Lemma 8.21) und wegen Lemma 8.22 die Dimension der beiden Vektorräume jeweils $n+1$ ist, müssen sie halt auch übereinstimmen. □

8.5 Knoteneinfügen

Definition 8.23 *Eine Knotenfolge $T^* = T_{m,n^*}$ heißt* Verfeinerung *von $T = T_{m,n}$, falls*

1. $t_0^* = t_0, \ldots, t_m^* = t_m$,
2. $t_{n^*+1}^* = t_{n+1}, \ldots, t_{n^*+m+1}^* = t_{n+m+1}$,
3. *es gibt $\tau : \{m+1, \ldots, n-1\} \to \{m+1, \ldots, n^*-1\}$ monoton steigend, so daß $t_{\tau(j)}^* = t_j$, $j = m+1, \ldots, n-1$.*

Formal schreiben wir das als $T \subseteq T^*$.[113]

Da Polynome unendlich oft differenzierbar sind, insbesondere an "zusätzlichen" Knoten, ist $\mathbb{S}_m(T) \subseteq \mathbb{S}_m(T^*)$, wann immer $T \subseteq T^*$. Anders gesagt, für jeden Vektor $\boldsymbol{d} = (\boldsymbol{d}_0, \ldots, \boldsymbol{d}_n)$ gibt es einen Vektor $\boldsymbol{d}^* = (\boldsymbol{d}_0^*, \ldots, \boldsymbol{d}_{n^*}^*)$, so daß

$$N_{m,T}\,\boldsymbol{d} = \underbrace{\sum_{j=0}^{n} \boldsymbol{d}_j\, N_j^m(\cdot|T)}_{\in \mathbb{S}_m(T)} = \underbrace{\sum_{j=0}^{n^*} \boldsymbol{d}_j^*\, N_j^m(\cdot|T^*)}_{\in \mathbb{S}_m(T^*)} = N_{m,T^*}\,\boldsymbol{d}^*. \tag{8.32}$$

[113] Die obigen Bedingungen bedeuten eine "$\subset$"–Relation, die Vielfachheiten berücksichtigt.

Die Preisfrage ist natürlich: "Wie bestimmt man $\boldsymbol{d}^*$?". Nun, im entscheidenden Fall des Einfügens *eines* Knotens (und natürlich läßt sich jede Verfeinerung einer Knotenfolge darauf zurückführen, indem man Knoten für Knoten einfügt) kann man das Verfahren angeben.
Sei also, für ein $j \in \{m, \ldots, n-1\}$

$$t_0 \leq \cdots \leq t_j \leq t^* \leq t_{j+1} \leq \cdots \leq t_{n+m+1},$$

und

$$T^* = \left\{t_k^* : k = 0, \ldots, n+m+2\right\}, \qquad t_k^* = \begin{cases} t_k & k = 0, \ldots, j, \\ t^* & k = j+1, \\ t_{k-1} & k = j+2, \ldots, n+m+2. \end{cases} \tag{8.33}$$

Der folgende Satz wird normalerweise Boehm[114] zugeschrieben, alternativ spricht man beim Knoteneinfügen auch gerne vom *Oslo–Algorithmus*[115].

Satz 8.24 (Knoteneinfügen) *Sei T^* eine Verfeinerung der Knotenfolge $T = T_{m,n}$ gemäß (8.33). Dann ergibt sich das Kontrollpolygon $\boldsymbol{d}^*$ als*

$$\boldsymbol{d}_k^* = \begin{cases} \boldsymbol{d}_k, & k = 0, \ldots, j-m, \\ u_0\left(t^* | I_k^m\right) \boldsymbol{d}_{k-1} + u_1\left(t^* | I_k^m\right) \boldsymbol{d}_k, & k = j-m+1, \ldots, j, \\ \boldsymbol{d}_{k-1}, & k = j+1, \ldots, n+1. \end{cases} \tag{8.34}$$

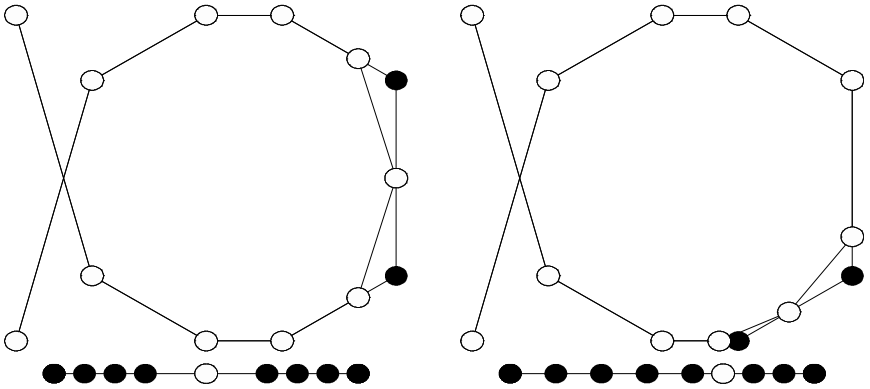


Abbildung 8.9: Zwei Beispiele für Knoteneinfügen bei kubischen Splines, $m = 3$.

Beweis: Übungsaufgaben. □

Übung 8.4 Beweisen Sie die *Leibniz–Formel* für dividierte Differenzen:

$$[x_0, \ldots, x_n]\,(fg) = \sum_{j=0}^{n} [x_0, \ldots, x_j]\, f \;\; [x_j, \ldots, x_n]\, g. \tag{8.35}$$

[114] Wer ihn nicht kennt, hat auch nicht so viel verpasst.
[115] Das Gebiet ist noch jung genug, um Prioritätenkonflikte zu erlauben.

Übung 8.5 Die *abgebrochene Potenz* $(\cdot - y)_+^n$ ist definiert als

$$(x-y)_+^n = \begin{cases} 0, & x \le y, \\ (x-y)^n, & x > y. \end{cases}$$

Zeigen Sie: Zu einer Kontenfolge $T = T_{m,n}$ ergibt sich der j–te B–Spline als

$$N_j^m(x|T) = [t_j, \ldots, t_{j+m+1}]\,(\cdot - x)_+^m, \qquad j = 0, \ldots, n,$$

Hinweis: Verwenden Sie die Leibniz–Formel für dividierte Differenzen, (8.35), um nachzuweisen, daß die rechte Seite des obigen Ausdrucks die Rekursionsformel für B–Splines erfüllt.

Übung 8.6 Es sei T^* die Knotenfolge, die entsteht, wenn der Knoten $t_j \le t^* < t_{j+1}$ in $T = T_{m,n}$ eingefügt wird. Zeigen Sie, daß dann

$$N_k^m(\cdot|T) = \begin{cases} N_k^m(\cdot|T^*), & k = 0, \ldots, j-m, \\ u_1(t^*|J_k^m)\, N_k^m(\cdot|T^*) + u_0(t^*|J_k^m)\, N_{k+1}^m(\cdot|T^*), & k = j-m+1, \ldots, j, \\ N_{k+1}^m(\cdot|T^*), & k = j+1, \ldots, n, \end{cases}$$

mit $J_k^m = \left[t_k^*, t_{k+m}^*\right]$, und folgern Sie daraus die Formel zum Knoteneinfügen.
Hinweis: Verwenden Sie die Darstellung des B–Splines als dividierte Differenz von abgebrochenen Potenzen.

8.6 Die Schoenberg–Whitney–Bedingung

So, jetzt machen wir uns also an den Beweis des Interpolationssatzes von Schoenberg und Whitney, also Teil 1 von Satz 8.6. Dazu bezeichnen wir mit

$$N_m(\mathscr{X} \mid T) = [N_k^m(x_j|T) \;:\; j,k = 0, \ldots, n]$$

die *Kollokationsmatrix* des Interpolationsproblems, das dann die Form

$$N_m(\mathscr{X} \mid T)\; \boldsymbol{d} = \boldsymbol{f}$$

annimmt und genau dann eindeutig lösbar ist, wenn $\det N_m(\mathscr{X} \mid T) \neq 0$. Eine Richtung von Satz 8.6 ist jetzt recht einfach.

Proposition 8.25 *Sei $m \in \mathbb{N}_0$ und $T = T_{m,n}$ eine Knotenfolge. Ist das Spline–Interpolationsproblem an Stellen $x_0, \ldots, x_n$ immer eindeutig lösbar, so ist $t_j < x_j < t_{j+m+1}$, $j = 0, \ldots, n$.*

Beweis: Nehmen wir an, es gäbe ein j, so daß die Schoenberg–Whitney–Bedingung (8.7), also $t_j < x_j < t_{j+m+1}$, *verletzt* ist. Dann ist entweder $x_j \le t_j$ oder $x_j \ge t_{j+m+1}$. Im ersten Fall gilt, da der Träger von $N_k^m(\cdot|T)$ das Intervall $[t_k, t_{k+m+1}]$ ist, die Beziehung

$$N_k^m(x_j|T) = 0, \qquad k = j, \ldots, n,$$

also haben die ersten $j+1$ Zeilen der Kollokationsmatrix die Form

$$\begin{bmatrix} N_0(x_0|T) & \dots & N_{j-1}(x_0|T) & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ N_0(x_j|T) & \dots & N_{j-1}(x_j|T) & 0 & \dots & 0 \end{bmatrix}$$

und sind damit linear abhängig, also ist $\det N_m(\mathscr{X}|T) = 0$. Im anderen Fall, $x_j \geq t_{j+m+1}$, ist

$$N_k^m(x_j|T) = 0, \qquad k = 0, \dots, j,$$

und, da $x_j \leq x_l$, $j \leq l$, gilt natürlich auch

$$N_k^m(x_l|T) = 0, \qquad k = 0, \dots, j,\ l = j, \dots, n.$$

Damit haben die ersten $j+1$ Spalten von $N_m(\mathscr{X}|T)$ die Form

$$\begin{bmatrix} N_0^m(x_0|T) & \dots & N_j^m(x_0|T) \\ \vdots & \ddots & \vdots \\ N_0^m(x_{j-1}|T) & \dots & N_j^m(x_{j-1}|T) \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix},$$

sind also ebenfalls linear abhängig. □

Bemerkung 8.26 *Die Idee dieses Beweises läßt sich auf eine viel allgemeinere Situation anwenden. Sieht man nämlich mal genau hin, dann haben wir nur ausgenutzt, daß die B–Splines einen wohldefinierten kompakten Träger haben, und sonst nichts.*

Die Umkehrung ist etwas aufwendiger, gibt aber gleichzeitig auch mehr Information über Eigenschaften der Splinefunktionen.

Definition 8.27 *Sei* $d = (d_0, \dots, d_n) \in \mathbb{R}^{n+1}$.[116] *Eine Teilmenge* $\Sigma = \{\sigma_1, \dots, \sigma_k\} \subset \{0, \dots, n\}$ *heißt* Signatur *bezüglich* d*, falls*

$$d_{\sigma_l} d_{\sigma_{l+1}} < 0, \qquad l = 0, \dots, k-1. \tag{8.36}$$

Mit $\Sigma(d)$ *bezeichnen wir die Menge aller Signaturen für* d *und definieren die Anzahl der echten Vorzeichenwechsel von* d*, in Zeichen* $S(d)$*, als*

$$S(d) = \max_{\Sigma \in \Sigma(d)} \#\Sigma.$$

Wir zeigen zuerst, daß die Zahl der Vorzeichenwechsel durch Knoteneinfügen nicht vergrößert wird.

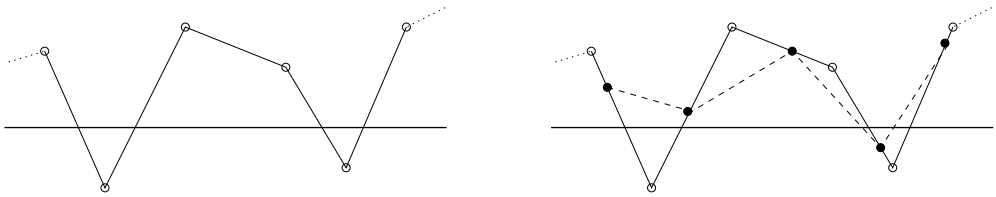


Abbildung 8.10: Die Beweisidee von Lemma 8.28. Die Variation des gestrichelten Polygons auf der rechten Seite, also $S(d^*)$, ist kleiner als die Variation des durchgezogenen Polygons (es werden ja zusätzliche Ecken dazwischengeschoben) mit schwarzen und weißen Ecken, bei dem man wiederum die schwarzen "Ecken" vergessen kann, weil sie auf Verbindungsgeraden liegen.

Lemma 8.28 *Ist T^* eine Verfeinerung von $T = T_{n,m}$ und ist $d^* \in \mathbb{R}^{1\times n^*}$ der durch Knoteneinfügen aus d enstandene Koeffizientenvektor, dann gilt*

$$S(d^*) \leq S(d).$$

Beweis: Es genügt natürlich, sich auf Verfeinerungen zu beschränken, die nur einen Knoten einfügen, allgemeine Verfeinerungen ergeben sich durch Iteration. Sei also

$$t_0 \leq \cdots \leq t_j \leq t^* \leq t_{j+1} \leq \cdots \leq t_{n+m+1}$$

und $T^* = T \cup \{t^*\}$. Nach Satz 8.24 ergeben sich die Koeffizienten als

$$d_k^* = \begin{cases} d_k & k = 0, \ldots, j-m, \\ u_0(t^*|I_k^m)\, d_{k-1} + u_1(t^*|I_k^m)\, d_k & k = j-m+1, \ldots, j, \\ d_{k-1} & k = j+1, \ldots, n+1. \end{cases}$$

Sei

$$d' = \left(d_0, \ldots, d_{j-m}, d^*_{j-m+1}, d_{j-m+1}, d^*_{j-m+2}, \ldots, d_{j-1}, d^*_j, d_j, d_{j+1}, \ldots, d_n\right).$$

Da d_k^* eine *Konvexkombination* von d_{k-1} und d_k ist, $k = j-m+1, \ldots, j$, ist

$$S(d_{k-1}, d_k^*, d_k) = S(d_{k-1}, d_k),$$

also $S(d') = S(d)$. Andererseits ist

$$d^* = \left(d_0, \ldots, d_{j-m}, d^*_{j-m+1}, \ldots, d^*_j, d_j, d_{j+1}, \ldots d_n\right)$$

eine Teilfolge von d' und damit ist $S(d^*) \leq S(d') = S(d)$. □

Definition 8.29 *Ein Punkt $x \in \mathbb{R}$ heißt* isolierte Nullstelle *einer Funktion $f \in C(\mathbb{R})$ falls $f(x) = 0$ und es für alle kompakten Intervalle $I \ni x$ mit nichtleerem Inneren einen Punkt $y \in I$ gibt, so daß $f(y) \neq 0$.*

[116] Wir betrachten jetzt also skalare Splinekurven, den Fall $d = 1$.

Bemerkung 8.30 *Isolierte Nullstellen müssen nicht einfach sein: $f(x) = x^{2n}$ hat eine $2n$–fache isolierte Nullstelle an $x = 0$. Umgekehrt hat die abgebrochene Potenz*[117] $(\cdot - x_0)_+^n$ *zwar jede Menge Nulstellen, nämlich $(-\infty, x_0]$, aber keine davon ist isoliert.*

Lemma 8.31 *Sei $k \geq 0$, $T = T_{m,n}$ eine Knotenfolge und $d \in \mathbb{R}^{n+1}$ ein Kontrollvektor so daß $d_l = 0$, $l \notin \{j, \ldots, j+k\}$ und so, daß die Splinefunktion $N_{m,T}d$ auf keinem Teilintervall von $I := [t_j, t_{j+k+m+1}]$ verschwindet. Dann hat $N_{m,T}d$ höchstens k isolierte Nullstellen in I.*

Beweis: Da $N_{m,T}\, d|_J \in \Pi_m$, $J = [t_\ell, t_{\ell+1}]$, $\ell = j, \ldots, j+k+m$, kann diese Funktion nur entweder vollständig verschwinden oder (im Inneren) isolierte Nullstellen haben. Da der Spline außerhalb von I verschwindet, sind die Nullstellen am Rand nicht isoliert!
Nach Lemma 8.14 ist $N_{m,T}d(x) = 0$, falls $x \notin I$. Sei also x^* eine Nullstelle von $N_{m,T}d$ im Inneren von I. Falls $x \notin T$ fügen wir x^* als m–fachen Knoten in die Knotenfolge T ein, falls x mit einem μ–fachen Knoten ($\mu \leq m$) übereinstimmt, dann erhöhen wir dessen Vielfachheit auf m. Dies liefert eine Knotenfolge T^* mit zugehörigem Kontrollvektor d^*. Nun ist aber

$$\#\left\{x \in I^\circ : N_{m,T}d(x) = 0\right\} = \#\left\{x \in I^\circ : N_{m,T^*}d^*(x) = 0\right\} \leq S\left(d^*\right) \leq S\left(d\right) \leq k.$$

Außerdem kann eine isolierte Nullstelle durch Knoteneinfügen offensichtlich nur dann entstehen, wenn vorher ein Vorzeichenwechsel da war. □

Beweis von Satz 8.6, "⇐": Nehmen wir an, es gäbe einen Vektor $0 \neq d \in \mathbb{R}^{n+1}$, so daß $N_m\left(\mathscr{X}|T\right)d = 0$, das heißt, daß $N_{m,T}\left(x_j|T\right) = 0$, $j = 0, \ldots, n$. Ist nun für ein $j \in \{m, \ldots, n\}$

$$0 = N_{m,T}d|_{I_j} = \sum_{k=j-m}^{j} d_k N_k^m\left(\cdot|T\right),$$

dann ist nach dem Algorithmus von de Boor

$$d_{j-m} = \cdots = d_j = 0$$

und $N_{m,T}d$ zerfällt in

$$N_{m,T}d = \underbrace{\sum_{k=0}^{j-m-1} d_k N_k^m\left(\cdot|T\right)}_{=:f_1} + \underbrace{\sum_{k=j+1}^{n} d_k N_k^m\left(\cdot|T\right)}_{=:f_2},$$

die die disjunkten Träger $[t_0, t_j]$ und $[t_{j+1}, t_{n+m+1}]$ haben. Auf diese Art und Weise können wir alle Teilintervalle, auf denen der Spline identisch verschwindet, entfernen und schließlich annehmen, daß wir nur noch *isolierte* Nullstellen haben: Gäbe es nämlich ein nichttriviales Teilintervall von irgendeinem I_j, auf dem der Spline verschwindet, dann müßte er, da er ja auf I_j ein Polynom ist, auf ganz I_j verschwinden.

Also können wir einen Kontrollvektor d finden, so daß $N_m\left(\mathscr{X}|T\right)\ d = 0$, und es j und k gibt, so daß $d_l = 0$, $l \notin \{j, \ldots, j+k\}$ und $N_{m,T}d$ hat im offenen Intervall $(t_j, t_{j+k+m+1})$ nur isolierte Nullstellen. Nun liegen aber in diesem Intervall die Interpolationspunkte $x_j, \ldots, x_{j+k}$, also müßte $N_{m,T}d$ dort $k+1$ isolierte Nullstellen haben, was aber Lemma 8.31 widerspricht. Also ist die Matrix $N_m\left(\mathscr{X}\,|\,T\right)$ nichtsingulär. □

[117] Bei de Boor: "Stutzfunktion" – klingt auch nicht besser.

8.7 Totale Nichtnegativität der Kollokationsmatrix

Beweis von Satz 8.6, 2: Sei t_j ein Knoten der Vielfachheit $< m$ und $t_j \le t^* \le t_{j+1}$. Nach Satz 8.24 ist dann für die durch Einfügen von t^* entstandene Knotenfolge T^*

$$N_k^m(\cdot|T) = \sum_{\ell=0}^{n+1} \alpha_{k,\ell} N_\ell^m(\cdot|T^*), \tag{8.37}$$

wobei

$$\alpha_{k,\ell} = \begin{cases} \delta_{k\ell} & \ell = 0,\dots,k-m, \\ u_0(t^*|I_\ell^m)\,\delta_{k,\ell-1} + u_1(t^*|I_\ell^m)\,\delta_{k\ell} & \ell = k-m+1,\dots,k, \\ \delta_{k,\ell-1} & \ell = k+1,\dots,n+1. \end{cases}$$

Insbesondere gilt[118], daß

$$\alpha_{k,\ell} \neq 0 \qquad \Longrightarrow \qquad \ell \in \{k, k+1\}. \tag{8.38}$$

Wir setzen

$$\mathscr{X}_I = \{x_i \,:\, i \in I\}, \qquad I \subseteq \{0,\dots,n\},$$

und definieren die *Spaltenvektoren*

$$N_j^m(\mathscr{X}_I|T) := \left[N_j^m(x_i|T) : i \in I\right], \qquad j = 0,\dots,n, \quad I \subseteq \{0,\dots,n\}.$$

Dann ist, für $I, J \subset \{0,\dots,n\}$, $\#I = \#J = k$,[119]

$$\begin{aligned}
\det N_m(\mathscr{X}|T)(I,J) &= \det\left[N_j^m(\mathscr{X}_I|T) : j \in J\right] \\
&= \det\left[N_{j_1}^m(\mathscr{X}_I|T), N_j^m(\mathscr{X}_I|T) : j \in J \setminus \{j_1\}\right] \\
&= \det\left[\sum_{\ell_1=0}^{n+1} \alpha_{j_1,\ell_1} N_{\ell_1}^m(\mathscr{X}_I|T^*), N_j^m(\mathscr{X}_I|T) : j \in J \setminus \{j_1\}\right] \\
&= \sum_{\ell_1=0}^{n+1} \alpha_{j_1,\ell_1} \det\left[N_{\ell_1}^m(\mathscr{X}_I|T^*), N_j^m(\mathscr{X}_I|T) : j \in J \setminus \{j_1\}\right] \\
&\;\;\vdots \\
&= \sum_{\ell_1=0}^{n+1} \alpha_{j_1,\ell_1} \cdots \sum_{\ell_k=0}^{n+1} \alpha_{j_k,\ell_k} \det\left[N_r^m(\mathscr{X}_I|T^*) : r = \ell_1,\dots,\ell_k\right] \\
&= \sum_{\ell_1=0}^{n+1} \cdots \sum_{\ell_k=0}^{n+1} \alpha_{j_1,\ell_1} \cdots \alpha_{j_k,\ell_k} \det N_m(\mathscr{X}|T^*)(I,\{\ell_1,\dots,\ell_k\}).
\end{aligned}$$

[118] Wegen der beteiligten $\delta_{k,\ell}$ und $\delta_{k,\ell-1}$.

[119] Zur Erinnerung: $N_m(\mathscr{X}|T)$ ist die Kollokationsmatrix und für eine Matrix A bezeichnet $A(I,J)$ die Teilmatrix, die durch die mit I bzw. J indizierten Zeilen bzw. Spalten gebildet wird.

Da, nach (8.38), $\alpha_{j_r,\ell_r} = 0$ falls $\ell_r \notin \{j_r, j_r+1\}$, und da[120] $j_1 < \cdots < j_k$, müssen die Elemente jeder Indexmenge $L = \{\ell_1, \ldots, \ell_k\}$ in der obigen Summe auf alle Fälle die Bedingung $\ell_1 \leq \cdots \leq \ell_k$ erfüllen. Tritt aber irgendwo Gleichheit ein, d.h. ist $\ell_r = \ell_{r+1}$, dann enthält die Matrix $N_m(\mathscr{X}|T^*)(I,L)$ *zwei identische* Spalten und damit ist die Determinante gleich Null. Damit läuft die Summe über alle $\ell_1 < \cdots < \ell_k$, also über alle $L \subseteq \{0, \ldots, n+1\}$[121] mit $\#L = k$. Mit anderen Worten, für

$$\alpha(J,L) := \prod_{r=1}^{k} \alpha_{j_r,\ell_r} \geq 0, \qquad J \subseteq \{0,\ldots,n\}, \quad L \subseteq \{0,\ldots,n+1\}, \quad \#J = \#L,$$

haben wir, daß

$$\det N_m(\mathscr{X}|T)(I,J) = \sum_{\#L=k} \alpha(J,L) \det N_m(\mathscr{X}|T^*)(I,L). \tag{8.39}$$

Fügen wir nun noch einen Knoten ein, dann ergibt sich enstprechend für die neue Knotenfolge T^{**}

$$\begin{aligned}
\det N_m(\mathscr{X}|T)(I,J) &= \sum_{\#L=k} \alpha(J,L) \det N_m(\mathscr{X}|T^*)(I,L) \\
&= \sum_{\#L=k} \alpha(J,L) \sum_{\#L'=k} \alpha(L,L') \det N_m(\mathscr{X}|T^{**})(I,L') \\
&= \sum_{\#L'=k} \underbrace{\sum_{\#L=k} \alpha(J,L)\alpha(L,L')}_{=:\alpha^2(J,L')} \det N_m(\mathscr{X}|T^{**})(I,L').
\end{aligned}$$

In diesem Sinne haben wir, für beliebiges $N \in \mathbb{N}$, nach Einfügen von N Knoten[122], für das resultierende T^{N*},

$$\det N_m(\mathscr{X}|T)(I,J) = \sum_{\#L=k} \alpha^N(J,L)\ \det N_m\left(\mathscr{X}|T^{N*}\right)(I,L), \tag{8.40}$$

wobei

$$\alpha^N(J,L) = \sum_{\#L_1=k} \cdots \sum_{\#L_{N-1}=k} \prod_{r=1}^{N+1} \alpha(L_r, L_{r+1}), \qquad L_r \subset \{0,\ldots,n+r\},\ r=1,\ldots,N. \tag{8.41}$$

Wir wählen nun T^{N*}[123] so dicht, daß zwischen je zwei Interpolationspunkten x_j und x_{j+1} mindestens $m+1$ Knoten liegen und daher, für $k = 0, \ldots, n$,

$$N_j^m(x_k|T^*) \neq 0 \quad \Rightarrow \quad N_j^m(x_\ell|T^*) = 0, \qquad \ell \neq k.$$

[120] Dies ist die kanonische Annahme, daß J *aufsteigend* numeriert ist

[121] Achtung: Der Index kann jetzt auch $n+1$ werden!

[122] Es genügt, wenn diese Knoten einfach, also $t_1^* < t_2^* < \cdots < t_N^*$ und neu, also $t_j^* \notin T$, sind.

[123] Genauer, die eingefügten Knoten.

Also hat *jede* Spalte *jeder* der (oberen Dreiecks-) Matrizen $N_m\left(\mathscr{X}|T^{N*}\right)(I,L)$, $L \subset \{0,\ldots,n+N\}$, höchstens einen von Null verschiedenen Eintrag. Liegt wenigstens einer davon nicht auf der Diagonale, dann ist $\det N_m\left(\mathscr{X}|T^{N*}\right)(I,L) = 0$, andernfalls ergibt sich

$$\det N_m\left(\mathscr{X}|T^{N*}\right)(I,L) = \prod_{r=1}^{k} \underbrace{N_{\ell_r}^m\left(x_{i_r}|T^*\right)}_{>0} > 0.$$

Setzt man das in (8.40) und berücksichtigt man (8.41), dann heißt das gerade, daß $\det N_m\left(\mathscr{X}\right)(I,J) > 0$. □

Πόλεμος πατήρ πάντων
Der Krieg ist der Vater aller Dinge

Heraklit, Frag. 53

8.8 Eine kurze Geschichte der Splines

Auch wenn man gerne die Arbeiten [40, 41] von Schoenberg aus dem Jahr 1946 als die "Geburtsstunde" der Splines bezeichnet, waren die B–Splines "wahrscheinlich Hermite und sicherlich Peano" (Schoenberg in [42]) bekannt. Will man noch weiter in der Geschichte zurückgehen, so kann man sich auf die *exponential Euler splines* berufen, die bereits 1755 von Euler, aber natürlich nicht unter diesem Namen, untersucht wurden. Eine Zusammenfassung der "Ur- und Frühgeschichte" der Splines gibt beispielsweise die Arbeit von Butzer, Schmidt und Stark [8].

Doch kehren wir wieder zurück zu Isaac J.[124] Schoenberg, einem der profiliertesten und vielseitigsten angewandten und nicht nur angewandten Mathematiker dieses Jahrhunderts. Im Rahmen seiner Tätigkeit im War Department at Aberdeen Proving Ground während des zweiten Weltkriegs begann er, ausgehend von Approximationsproblemen, mit der Untersuchung von Splines. Es ist einer der wenigen positiven Aspekte des Kriegs, daß zur selben Zeit auch Haskell B. Curry, ansonsten eher der Algebra und abstrakten Logik zugetan, von diesem Ansatz Kenntnis bekam und ihn gemeinsam mit Schoenberg weiterentwickelte. Eine Kopie eines Briefes, in dem Curry von diesen Entwicklungen erzählt, findet sich im Buch von Micchelli [25] (das sich beispielsweise mit den Themen Subdivision, Splines mit verallgemeinerten Differenzierbarkeitseigenschaften, multivariaten Splines und Blossoming befaßt und schon deswegen sehr empfehlenswert ist). Trotzdem beschäftigte sich auch Schoenberg zuerst einmal lange Zeit mit anderen Dingen, so daß die berühmte Arbeit von Curry und Schoenberg [12] erst mit fast 20–jähriger Verspätung im Jahre 1966 erschien. Einer der Gründe, warum Schoenberg seine Resultate über Splines wieder "aus der Schublade" holte, war wohl die Tatsache, daß inzwischen auch andere Mathematiker, allen voran Carl de Boor, die Splines entdeckt hatten und so Schoenbergs Interesse neu weckten. Natürlich bestand einer der Gründe für das aufkeimende allgemeine Interesse an Splines im Bedarf an "guten" Typen von Kurven im Zusammenhang mit den CAD–Systemen und den numerischen Berechnungen, die durch die Entwicklung der Computertechnik in ganz neuen Größenordnungen durchgeführt wurden. Auf der anderen Seite waren und sind Splines auch ein wichtiges theoretisches Hilfsmittel bei der Untersuchung von *n–ten Weiten*, wo sie eine wichtige Rolle als Minimallösungen bestimmter Funktionale spielen. Inzwischen ist die Literatur zum Thema "Splines" gigantisch. Die beste, umfangreichste und via Internet zugängliche Bibliographie stammt von de Boor und kann beispielsweise über seine WWW–Homepage `http://www.cs.wisc.edu/~deboor` abgerufen werden.

In den meisten "klassischen" Lehrbüchern über Numerische Mathematik werden Splines (oft nur kubisch, das heißt von der Ordnung $m = 3$) als Lösung eines Interpolationsproblems an den *einfachen Knoten* $t_{m+1}, \ldots, t_n$ eingeführt und beschrieben. Nicht, daß das irgendetwas

[124] Jacob

schlechtes wäre, denn gerade in diesem Zusammenhang zeigen die Splines ein den Polynomen weit überlegenes Verhalten, da sie das oftmals katastrophale Oszillationsverhalten der Interpolationspolynome nicht mitmachen. Auf alle Fälle hat man es aber in diesem Rahmen wegen der einfachen Knoten stets mit Funktionen aus C^{m-1} zu tun. Eine Basis des zugehörigen Splineraums ist dann schnell angegeben: sie besteht aus allen Polynomen vom Grad m sowie den *abgebrochenen Potenzen*

$$(t-t_i)_+^m = \begin{cases} (t-t_i)^m & t \geq t_i, \\ 0 & t < t_i, \end{cases} \quad i = m+1, \ldots, n-1.$$

Klar – diese Funktionen sind alle linear unabhängig, stückweise polynomial und $m-1$ mal stetig differenzierbar. Damit haben wir also $m+1$ Polynome und $n-m-1$ abgebrochene Potenzen, also n Basisfunktionen, und da wir, dank der B–Splines, wissen, daß der Splineraum Dimension n hat, sind diese Funktionen ebenfalls eine Basis. Allerdings eine Basis mit Nachteilen: die Basisfunktionen haben unendlichen Träger! Deswegen war man interessiert an einer Basis des Splineraums dergestalt, daß alle Basisfunktionen *möglichst kleinen* Träger haben – und das führte zu den B–Splines. Daß diese Funktionen dann auch noch über eine Rekursionsformel verknüpft sind, wie Cox [11] und de Boor [4] zeigten, macht sie nur noch sympathischer. Zudem haben die Basisfunktionen mit kompaktem Träger noch einen weiteren Vorteil, den bereits Schoenberg weidlich ausnutzte: da zur Berechnung von $\boldsymbol{N}^m(t)$ stets nur die m Knoten "links von t" und die m Knoten "rechts von t", sowie die $m+1$ zugehörigen Kontrollpunkte verwendet werden, gibt es überhaupt keine Schwierigkeiten, auch *unendliche* Knotenfolgen t_i, $i \in \mathbb{Z}$, mit unendlichen Kontrollpolygonen $\boldsymbol{d}_i$, $i \in \mathbb{Z}$, zu betrachten. Und das ist mit einer Basis aus Polynomen und abgebrochenen Potenzen nun doch etwas unschön, da hier an jedem Punkt eine *unendliche* Anzahl von Basisfunktionen beiträgt. Aber besser noch: sind die Knoten *gleichverteilt*, also z.B. $t_i = i$, $i \in \mathbb{Z}$, (das sind genau die *Cardinal B–Splines* von Schoenberg [42]), dann sind die B–Splines lediglich verschobene Kopien voneinander, genauer

$$N_i^m(x) = N_0^m(x-i), \qquad i \in \mathbb{Z},$$

und man hat eigentlich nur *einen* B–Spline.

Aus den 70er und 80er Jahren gibt es eine Vielzahl von Arbeiten über Splines, die für einige Zeit ein richtiges "Modethema" waren. Einen gewissen Überblick aus unterschiedlichen Perspektiven und von unterschiedlichen Zugängen her bieten beispielsweise die Bücher von de Boor [5], Schumaker [43] oder Nürnberger [28]. Auf alle Fälle fand man eine Vielzahl von Problemen und Gebieten, wo sich die Splines als mehr oder weniger nützlich erwiesen.

Trotzdem geschah noch einmal das Wunder, daß auf einem eigentlich "abgegrasten" Feld, das die (univariaten) Splines in den 80er Jahren mit Sicherheit waren, noch einmal zarte Blüten (engl. "*blossoms*") sprossen. Es war überraschenderweise wieder de Casteljau, der zuerst auf die Zusammenhänge mit polaren Formen aufmerksam machte [9], wenn auch in sehr schwer lesbarer Form. Die Anwendung von "Blossoming" auf Splinekurven geht wohl auf Lyle Ramshaw [34] zurück (siehe auch und vor allem [35]). Die Darstellung in diesem Skript folgt in groben Zügen [37], die ihrerseits auf einer Arbeit von Hans–Peter Seidel[125] [45] basiert, welche im

[125] Hat nichts mit Gauß–Seidel zu tun. Und so selten ist der Name "Seidel" ja nun auch wieder nicht.

wesentlichen für meine Begeisterung für dieses Gebiet verantwortlich ist. Seidels Ziel war es übrigens, die neuen Einsichten, die das Blossoming gewährte, auch auf Flächen (generell, auf den multivariaten Fall) zu übertragen. Obwohl da einige Dinge ganz entschieden anders sind als bei den Kurven, gelang es doch, Splineflächen von hochgradiger Flexibilität zu konstruieren. Aber das ist eine andere Geschichte, die nur in einer Spezialvorlesung erzählt werden kann . . .

Also daß Rechnen ein fundament und grundt aller Künste ist / Dann ohne zal mag kein Musicus seinen Gesang / kein Geometer sein Mensur vollbringe / auch kein Astronomus den lauff des Himmels erkennen.

Adam Riese [36]

9 Nichtlineare Gleichungen

Das Thema dieses Kapitels ist die Lösung beliebiger, auch *nichtlinearer* Gleichungen der Form

$$F(x) = 0, \qquad F : \mathbb{R}^n \to \mathbb{R}^n,\ n \in \mathbb{N}, \tag{9.1}$$

wobei F natürlich gewissen Voraussetzungen zu genügen hat, und zwar mindestens stetig, meistens sogar differenzierbar sein soll. Daß die rechte Seite von (9.1) den Wert 0 hat, ist natürlich keine Einschränkung: Für beliebige rechte Seite $G(x)$ ist ja[126]

$$F(x) = G(x) \qquad \Longleftrightarrow \qquad (F - G)(x) = 0.$$

Beispiel 9.1 *Die Berechnung der* Quadratwurzel *einer rationalen Zahl $0 < y \in \mathbb{Q}$ war schon in der griechischen Antike ein wichtiges Problem und gelöst. Die dazugehörige nichtlineare Gleichung ist*

$$f(x) = 0, \qquad f(x) = x^2 - y,$$

und kann über die oft als Heron–Verfahren *bezeichnete Iteration*

$$x^{(0)} = y, \qquad x^{(k+1)} = \frac{1}{2}\left(x^{(k)} + \frac{y}{x^{(k)}}\right), \quad k \in \mathbb{N}_0, \tag{9.2}$$

gelöst werden, die[127] *sogar schon den Babyloniern bekannt war.*
Das Heron–Verfahren kann auch geometrisch interpretiert werden, nämlich als Konstruktion flächengleicher Rechtecke, die in der Grenze ein Quadrat ergeben, dessen Seitenlänge die gewünschte Wurzel ist – starten kann man ja mit dem Rechteck mit den Seitenlängen y und 1. Ein Konstruktionsschritt ist in Abb. 9.1 beschrieben.
Die Durchführung des "Iterationsschritts", das heißt die Berechnung der Seiten eines Rechtecks aus ihrer Summe und der Fläche des Rechtecks, ist als babylonischer Keilschrifttext überliefert,

[126] Das setzt natürlich voraus, daß auch G eine "gute" Funktion im Sinne eines funktionierenden numerischen Verfahrens ist.

[127] Siehe z.B. [16]

siehe [33, S. 125–127]. Laut [29] sind Keilschrifttexte wesentlich stärker algebraisch als geometrisch orientiert – ganz im Gegensatz zu den Rechenmeistern der Renaissance, siehe z.B. [22].

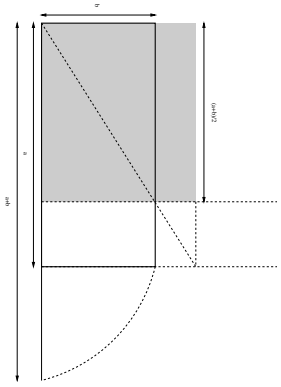


Abbildung 9.1: Ein Iterationsschritt (9.2) des Heron–Verfahrens geometrisch: Konstruiere zu vorgegebenen Seitenlängen a und b das flächengleiche Rechteck mit den Seitenlängen $\frac{a+b}{2}$ und $\frac{2ab}{a+b}$.

Eigentlich ist das alles aber noch viel einfacher mit der geometrischen Heuristik und Rechtfertigung des Heron–Verfahrens. Wir wollen ja ein Rechteck in ein flächengleiches Quadrat konvertieren. Hat dieses Rechteck nun Fläche y und eine bekannte Seite der Länge x, dann hat die andere Seite offensichtlich Länge y/x. Ist unser Rechteck kein Quadrat, dann ist x entweder zu kurz, und damit y/x zu lang oder x ist zu lang und y/x zu kurz. Ein besserer Wert als x liegt also immer zwischen x und y/x und könnte somit als

$$x^* = \alpha\, x + (1-\alpha)\,\frac{y}{x}, \qquad \alpha \in (0,1), \tag{9.3}$$

geschrieben werden. Es gibt sogar ein "magisches" α, das genau das gewünschte Quadrat liefert, aber das zu bestimmen ist nicht einfach. Also wählt man etwas universell passendes, nämlich $\alpha = \frac{1}{2}$ und schon ist man bei der Heron–Iteration! Sollte zufällig x so gewählt gewesen sein, daß wir schon ein Quadrat haben, dann liefert (9.3) *für jedes* α auch dieses x zurück – das ist eben die Fixpunkteigenschaft.

Übung 9.1 Zeigen Sie, daß das Heron–Verfahren immer gegen die Quadratwurzel von y konvergiert. Hinweis: Bestimmen Sie die Fixpunkte der Iteration.

9.1 Terminologie

Definition 9.2 *Ein iteratives Verfahren* $x^{(k)} \in \mathbb{R}^n$, $k \in \mathbb{N}_0$, *mit* $\lim_{k\to\infty} x^{(k)} = x$ *hat die* Konvergenzordnung $p \geq 1$, *bezüglich des* Fehlermaßes[128] $\varepsilon_k \geq 0$, $k \in \mathbb{N}$, $\lim_{k\to\infty} \varepsilon_k = 0$ *wenn*

[128] Normalerweise definiert durch Abschätzungen der Form $\varepsilon_k \geq \left\|x^{(k)} - x\right\|$.

es eine Konstante $C > 0$ gibt, so daß

$$\limsup_{k\to\infty} \frac{\varepsilon_{k+1}}{\varepsilon_k^p} = C. \tag{9.4}$$

Ist $p = 1$, so muß außerdem $C < 1$ sein.

Anschaulich heißt das, daß ε_k wir $C\,\rho^{p^k}$ für ein $0 < \rho < 1$ gegen Null konvergiert.

Definition 9.3 *Sei X ein metrischer Raum. Ein iteratives Verfahren $x^{(k)} \in X$, $k \in \mathbb{N}_0$, heißt* global konvergent*, wenn es für jeden Startwert $x^{(0)}$ konvergiert und* lokal konvergent*, wenn es eine offene Menge $U \subset X$ gibt, so daß das Verfahren für alle Startwerte $x^{(0)} \in U$ konvergiert.*

9.2 Das Bisektionsverfahren

Das einfachste Verfahren zur Nullstellenbestimmung ist das *Bisektionsverfahren*, oder *Einschlußverfahren*, das eine Nullstelle mit beliebiger Genauigkeit ermittelt, vorausgesetzt, man kennt Stellen, an der die Funktion einen positiven und einen negativen Wert hat. Sind also $x^{(0)}$ und $y^{(0)}$ so, daß[129]

$$f\left(x^{(0)}\right) < 0, \qquad f\left(y^{(0)}\right) > 0,$$

dann setzen wir, für $k \in \mathbb{N}_0$,

$$\begin{aligned} z &= \frac{x^{(k)} + y^{(k)}}{2} \\ x^{(k+1)} &= \begin{cases} x^{(k)} & f(z) > 0, \\ z & f(z) < 0, \end{cases} \\ y^{(k+1)} &= \begin{cases} z & f(z) > 0, \\ y^{(k)} & f(z) < 0. \end{cases} \end{aligned} \tag{9.5}$$

Ist zufällig $f(z) = 0$, dann haben wir ohnehin die Nullstelle gefunden.

Satz 9.4 *Ist $f \in C[a,b]$ und sind $x^{(0)}, y^{(0)} \in [a,b]$ so daß*

$$f\left(x^{(0)}\right) < 0, \qquad f\left(y^{(0)}\right) > 0,$$

dann konvergieren die Folgen $x^{(k)}, y^{(k)}$, $k \in \mathbb{N}_0$, definiert durch (9.5) von linearer Ordnung gegen eine Nullstelle x von f.

Bemerkung 9.5 *(Bisektionsverfahren)*

1. *Der große Vorteil des Bisektionsverfahrens ist seine Einfachheit: Es ist fast schon trivial zu programmieren und verlangt auch keine großen mathematischen Fähigkeiten, was es besonders populär macht.*

[129] Wäre f an einer der beiden Stellen $= 0$, dann können wir uns das Verfahren sparen!

```
%#
%# Bisekt.m
%# Bisektionsverfahren
%# Daten:
%#   f  Funktione
%#   x  Startwert - f(x) < 0
%#   y  Startwert - f(y) > 0
%#   t  Toleranz

function xx = Bisekt( f,x,y,t )
  while norm( x - y ) > t
    z = .5 * ( x + y );
    fz = feval( f,z );

    if fz > 0
      y = z;
    else
      x = z;
    end
  end
  xx = x;
%endfunction
```

Programm 9.1 `Bisekt.m`: Das Bisektionsverfahren.

2. *Die Konvergenzaussage gilt*[130] *nicht nur für Intervalle, sondern für beliebige konvexe metrische Räume, also insbesondere für Funktionen $f \in C(\mathbb{R}^n)$.*

3. *Man kann sogar zeigen, daß für die Nullstellensuche bei beliebigen stetigen Funktionen das Bisektionsverfahren* optimal *ist – es gibt keine bessere Konvergenzrate. Allerdings liegt das nicht daran, daß das Bisektionsverfahren so genial ist, sondern daran, daß die stetigen Funktionen so eine große Klasse bilden und daß man für "bessere" Verfahren immer eine hinreichend pathologische stetige Funktion finden kann, bei der Bisektion dann doch wieder besser abschneidet.*

4. *Der hauptsächliche Nachteil ist die relativ langsame, "nur" lineare Konvergenz, die im wesentlichen* unabhängig *von der Funktion ist. Auf der anderen Seite ist diese "Unabhängigkeit" aber auch genau der Grund, warum das Bisektionsverfahren optimal auf der Klasse der stetigen Funktionen ist.*

5. *Außerdem muß man zuerst einmal ein Intervall finden, in dem ein Vorzeichenwechsel stattfindet. Sind in diesem Intervall dann* mehrere *Nullstellen, wird es schwer, vorherzusagen, welche Nullstelle gefunden wird.*

Beweis von Satz 9.4: Wir können ohne Einschränkung annehmen, daß $x^{(0)} < y^{(0)}$ ist[131], und halten fest, daß das Bisekstionsverfahren ausgehend von $a_0 = x^{(0)}$ und $b_0 = y^{(0)}$

- entweder Intervalle $I_k = [a_k, b_k]$ bestimmt, so daß $g(a_k) < 0$, $g(b_k) > 0$ und $|b_k - a_k| = 2^{-k}(b-a)$,
- oder abbricht, weil $g(a_k) = 0$ oder $g(b_k) = 0$ (oder beides) für ein $k > 0$ ist.

Da g stetig ist, gibt es nach dem Zwischenwertsatz im ersten Fall[132] immer eine Nullstelle $t_k \in (a_k, b_k)$, und da

$$t_k - a_k \leq b_k - a_k \leq \frac{b-a}{2^k} \qquad \text{sowie} \qquad b_k - t_k \leq b_k - a_k \leq \frac{b-a}{2^k}$$

ist, konvergieren sogar beide Folgen gegen eine[133] Nullstelle. Die lineare Konvergenzordnung ergibt sich aus dem Fehlermaß $\varepsilon_k = |b_k - x^*|$ bzw. $\varepsilon_k = |a_k - x^*|$ als

$$\frac{|b_{k+1} - x^*|}{|b_k - x^*|} \sim \frac{|a_{k+1} - x^*|}{|a_k - x^*|} \sim \frac{1}{2}$$

ist. □

[130] Ohne echten Mehraufwand im Beweis.

[131] Andernfalls funktioniert der Beweis ganz genauso!

[132] Und nur der ist interessant, denn im zweiten Fall haben wir ja schon eine Nullstelle gefunden.

[133] Vorsicht: nicht **die**! Man denke nur an ein ganzes Nullstellenintervall im Inneren oder an die Nullstellen einer Funktion wie $x \sin \frac{1}{x}$ in der Nähe der Null.

9.3 Regula falsi und das Sekantenverfahren

Das nächste Verfahren zur Nullstellenbestimmung ist ein "Klassiker", nämlich die *regula falsi*. Der Name stammt von Fibonacci (Leonardo Pisano, 13. Jhdt.), der sie aus dem Arabischen als *Elchataym* übernahm und lateinisch als

> *regula duarum falsarum positionum*

bezeichnete. Bei ihm ist sie **die** Methode, mit der alle mathematischen Probleme gelöst werden können, siehe [46, Kap. 13, S. 447]. Zur Erklärung des Namens sagt laut [16, S. 245] der Rechenmeister Peter Bienewitz (1527)

> *Vnd heisst nit darum falsi dass*[134] *sie falsch und unrecht wehr, sunder, dass sie auss zweyen falschen vnd vnwahrhaftigen zalen und zweyen lügen die wahrhaftige vnd begehrte zal finden lernt.*

Typischerweise wurde die Regula Falsi in der Renaissance genutzt, um *lineare* Gleichungen zu lösen, was sie in einem Schritt tut – wir werden sie als Iterationsverfahren für beliebige nichtlineare Gleichungen verwenden. Vorher aber ein Beispiel aus [36]:

> *Item einer spricht / Gott grüß euch Gesellen alle dreissig. Antwortet einer / wann unser noch so viel und halb so viel weren / so weren unser dreissig. Die frag wie viel ihr gewesen*[135].

So, jetzt aber aus der Spaß und zurück zur Mathematik. Die Regula Falsi ist nämlich eine Verallgemeinerung des Bisektionsverfahrens: sind $x^{(k)}$ und $y^{(k)}$ zwei Punkte mit

$$f\left(x^{(k)}\right)\ f\left(y^{(k)}\right) < 0,$$

dann unterteilt man nicht am Mittelpunkt des Intervalls $\left[x^{(k)}, y^{(k)}\right]$, sondern an der Nullstelle z der Geraden ℓ mit

$$\ell\left(x^{(k)}\right) = f\left(x^{(k)}\right), \qquad \ell\left(y^{(k)}\right) = f\left(y^{(k)}\right).$$

Da

$$\ell\left((1-\alpha)\ x^{(k)} + \alpha\ y^{(k)}\right) = (1-\alpha)\ \ell\left(x^{(k)}\right) + \alpha\ \ell\left(y^{(k)}\right) = (1-\alpha)\ f\left(x^{(k)}\right) + \alpha\ f\left(y^{(k)}\right)$$

ist

$$z = (1-\alpha)\ x^{(k)} + \alpha\ y^{(k)}, \qquad \alpha = \frac{f\left(x^{(k)}\right)}{f\left(x^{(k)}\right) - f\left(y^{(k)}\right)}, \tag{9.6}$$

und der Iterationsschritt ist wie beim Bisektionsverfahren

$$x^{(k+1)} = \begin{cases} x^{(k)} & f\left(x^{(k)}\right)\ f(z) > 0, \\ z & f\left(x^{(k)}\right)\ f(z) < 0, \end{cases} \qquad y^{(k+1)} = \begin{cases} y^{(k)} & f\left(y^{(k)}\right)\ f(z) > 0, \\ z & f\left(y^{(k)}\right)\ f(z) < 0. \end{cases} \tag{9.7}$$

Übrigens wurde die Regula Falsi auch dazu benutzt, Gleichungen in *zwei* Unbekannten zu lösen. Hier ein Beispiel aus [36]:

---

[134] Was zeigt, wie progressiv die Rechtschreibreform 1999 wirklich war.

[135] Für die, die es nicht gleich begriffen haben: Zu lösen ist die Gleichung $\frac{5}{2}x = 30$ und tatsächlich kommt – nach Adam Riese – auch 12 heraus.

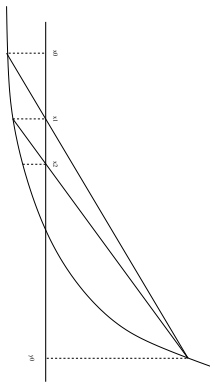


Abbildung 9.2: Geometrische Interpretation der Regula Falsi.

Item / zween wöllen ein Pferdt kaufen / Als A. vnd B. für 15. fl. Spricht A. zum B. gib mir deines gelts ein drittheil / so will ich meins darzu thun / vnd das Pferdt bezahlen. Spricht B. zum A. gib mir von deinem gelt ein viertheil / so wil ich mit meinem gelt hinzu das pferdt bezahlen. Nun frage ich / wie viel jeglicher in sonderheit gelts hab? [136]

Übung 9.2 Zeigen Sie, daß die Regula Falsi in einem reellen Intervall konvergiert und Konvergenzordnung 1 hat.

Allerdings erbt die Regula Falsi immer noch ein Problem des Bisektionsverfahrens: Um konvergieren zu können, brauchen wir Startwerte $x^{(0)}$ und $y^{(0)}$, an denen f verschiedene Vorzeichen hat. Das *Sekantenverfahren* umgeht dieses Problem, indem man zur Berechnung von $x^{(k+1)}$ die Gerade (Sekante!) verwendet, die f an den Stellen $x^{(k-1)}$ und $x^{(k)}$ interpoliert , die Regel lautet also

$$x^{(k+1)} = x^{(k)} - \frac{x^{(k)} - x^{(k-1)}}{f\left(x^{(k)}\right) - f\left(x^{(k-1)}\right)} f\left(x^{(k)}\right), \tag{9.8}$$

ihre Herleitung ist wie die von (9.6). Das Schöne am Sekantenverfahren ist nun, daß es lokal konvergent mit einer höheren Konvergenzordnung als 1 ist.

Satz 9.6 *Es sei $x \in \mathbb{R}$ eine* einfache *Nullstelle von $f \in C^2(U_x)$ für eine offene Umgebung U_x von x. Dann ist das Sekantenverfahren lokal konvergent gegen x und die Konvergenzordnung ist mindestens $p = \frac{1}{2}\left(1 + \sqrt{5}\right)$.*

Die Zahl $p = \frac{1}{2}\left(1 + \sqrt{5}\right)$ bezeichnet man als den *goldenen Schnitt* und ist der Grenzwert zweier aufeinanderfolgender *Fibonacci–Zahlen*. Der goldene Schnitt spielt als Proportion eine zentrale Rolle in Kunst und Architektur, aber auch in der Natur.

[136] Also $x + \frac{1}{3}y = \frac{1}{4}x + y = 15$ und damit $x = 10\frac{10}{11}$ und $y = 12\frac{3}{11}$. Die wesentlich interessantere Frage, wie man $\frac{10}{11}$ beziehungsweise $\frac{3}{11}$ einer Währung bestimmt oder erhält, wird von Adam Riese leider nicht beantwortet.

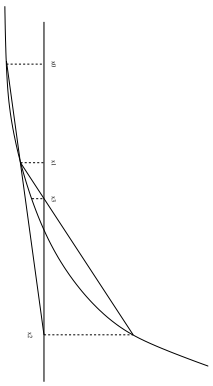


Abbildung 9.3: Geometrische Interpretation des Sekantenverfahrens.

Beweis: Für $\varepsilon > 0$ bezeichne

$$B_\varepsilon(x) = \{y \in \mathbb{R} \ : \ |y - x| < \varepsilon\}$$

die Kugel vom Radius ε um x. Wir wählen ε so klein, daß $B_\varepsilon(x) \subseteq U_x$; dann gilt für die Zahl

$$M_\varepsilon = \max_{y,y' \in B_\varepsilon(x)} \left| \frac{f''(y)}{2f'(y')} \right|,$$

daß

$$\lim_{\varepsilon \to 0} M_\varepsilon = \left| \frac{f''(x)}{2f'(x)} \right| < \infty.$$

Also gibt es ein ε_0, so daß für alle $\varepsilon < \varepsilon_0$

$$B_\varepsilon(x) \subseteq U_x \qquad \text{und} \qquad \varepsilon M_\varepsilon =: \gamma < 1 \qquad \text{und} \qquad 0 \notin f'\left(B_\varepsilon(x)\right).$$

Wir fixieren nun ein solches $\varepsilon < \varepsilon_0$. Dann ist aber x die einzige Nullstelle von f in $B_\varepsilon(x)$: Die Taylor–Formel um x für $y \in B_\varepsilon(x)$

$$f(y) = \underbrace{f(x)}_{=0} + (y - x)\, f'(x) + \frac{(y - x)^2}{2} f''(\xi), \qquad \xi \in [x, y] \subset B_\varepsilon(x),$$

liefert

$$f(y) = (y - x)\, f'(x) \left(1 + \frac{y - x}{2} \frac{f''(\xi)}{f'(x)}\right),$$

also

$$|f(y)| \geq |y - x|\, |f'(x)| \ \left(1 - \left| \frac{y - x}{2} \frac{f''(\xi)}{f'(x)} \right|\right) > |y - x|\, |f'(x)|\, (1 - \varepsilon M_\varepsilon) > 0.$$

Wir starten nun mit $x^{(0)}, x^{(1)} \in B_\varepsilon(x)$ und iterieren nach (9.8). Dann gilt für $k \in \mathbb{N}$ unter Verwendung der dividierten Differenzen, siehe [38, Definition 7.23], solange[137] $x^{(k-1)}, x^{(k)} \neq x$

$$\begin{aligned}
x^{(k+1)} - x &= x^{(k)} - \underbrace{\frac{x^{(k)} - x^{(k-1)}}{f\left(x^{(k)}\right) - f\left(x^{(k-1)}\right)}}_{=1/\left[x^{(k-1)}, x^{(k)}\right] f} f\left(x^{(k)}\right) - x = x^{(k)} - x - \frac{f\left(x^{(k)}\right)}{\left[x^{(k-1)}, x^{(k)}\right] f} \\
&= \left(x^{(k)} - x\right)\left(1 - \frac{f\left(x^{(k)}\right) - f(x)}{x^{(k)} - x} \, \frac{1}{\left[x^{(k-1)}, x^{(k)}\right] f}\right) \\
&= \left(x^{(k)} - x\right)\left(1 - \frac{\left[x^{(k)}, x\right] f}{\left[x^{(k-1)}, x^{(k)}\right] f}\right) \\
&= \frac{\left(x^{(k)} - x\right)\left(x^{(k-1)} - x\right)}{\left[x^{(k-1)}, x^{(k)}\right] f} \, \underbrace{\frac{\left[x^{(k-1)}, x^{(k)}\right] f - \left[x^{(k)}, x\right] f}{x^{(k-1)} - x}}_{=\left[x^{(k-1)}, x^{(k)}, x\right] f} \\
&= \left(x^{(k)} - x\right)\left(x^{(k-1)} - x\right) \frac{\left[x^{(k-1)}, x^{(k)}, x\right] f}{\left[x^{(k-1)}, x^{(k)}\right] f}
\end{aligned}$$

Sind nun $x^{(k-1)}, x^{(k)} \in B_\varepsilon(x)$, dann ist

$$\left[x^{(k-1)}, x^{(k)}\right] f = f'\left(\xi_1\right), \qquad \left[x^{(k-1)}, x^{(k)}, x\right] f = \frac{1}{2} f''\left(\xi_2\right), \qquad \xi_1, \xi_2 \in B_\varepsilon(x),$$

und damit ist

$$\left|x^{(k+1)} - x\right| = \underbrace{\left|x^{(k)} - x\right|}_{\leq \varepsilon} \underbrace{\left|x^{(k-1)} - x\right|}_{\leq \varepsilon} \underbrace{\left|\frac{f''\left(\xi_2\right)}{2f'\left(\xi_1\right)}\right|}_{\leq M_\varepsilon} \leq \varepsilon^2 M_\varepsilon < \varepsilon, \tag{9.9}$$

also ist auch $x^{(k+1)} \in B_\varepsilon(x)$. Außerdem ist

$$\left|x^{(k+1)} - x\right| \leq \varepsilon M_\varepsilon \left|x^{(k)} - x\right| \leq \left(\varepsilon M_\varepsilon\right)^2 \left|x^{(k-1)} - x\right| \leq \cdots \leq \gamma^{k+1} \underbrace{\left|x^{(0)} - x\right|}_{\leq \varepsilon},$$

also konvergiert die Folge $x^{(k)} \to x$ für $k \to \infty$.
Bleibt noch die Konvergenzordnung. Dazu können wir annehmen, daß $x^{(k)} \neq x$, $k \in \mathbb{N}_0$, denn sonst hätten wir ja Konvergenz nach endlich vielen Schritten. Wir definieren $E_k = M_\varepsilon \left|x^{(k)} - x\right|$, $k \in \mathbb{N}$, und erhalten aus (9.9), daß

$$E_{k+1} = M_\varepsilon \left|x^{(k+1)} - x\right| \leq M_\varepsilon^2 \left|x^{(k)} - x\right| \left|x^{(k-1)} - x\right| \leq E_k \, E_{k-1}.$$

Mit $E = \max\left\{E_0, E_1^{1/p}\right\}$ ist $E_k \leq E^{p^k}$ für $k = 0, 1$, und generell folgt für $k \geq 1$ unter Verwendung der Identität[138] $p^2 = p + 1$, daß

$$E_{k+1} \leq E_k \, E_{k-1} \leq E^{p^k + p^{k-1}} = E^{p^{k-1}(p+1)} = E^{p^{k-1} p^2} = E^{p^{k+1}}.$$

[137] Ansonsten sind wir sowieso fertig und haben die Nullstelle gefunden!
[138] Sie definiert den goldenen Schnitt.

Also ist

$$\left|x^{(k)} - x\right| = \frac{E_k}{M_\varepsilon} \leq \frac{E^{p^k}}{M_\varepsilon} =: \varepsilon_k$$

und damit

$$\frac{\varepsilon_{k+1}}{\varepsilon_k^p} = M_\varepsilon^{p-1}.$$

□

Zum Abschluß dieses Abschnitts noch eine "Gebrauchsanweisung" für die Regula Falsi, natürlich wieder von Adam Riese persönlich [36, S. 57,58]:

Regula Falsi oder Position.

Wirdt gefaßt von zweyen falschen zahlen / welche der auffgab nach / mit fleiß examinirt sollen werden / in massen das fragstück begeren ist / sagen sie der warheit zu viel / so bezeichne sie mit dem zeichen + plus / wo aber zu wenig / so beschreib sie mit dem zeichen − minus genannt. Als dann nimb ein lügen von der andern / was da bleibt / behalt für den theiler / multiplicir darnach im Creuz ein falsche zahl mit der andern lügen / nimb eins vom andern / vnd das da bleibt theil ab mit fürgemachtem theiler / so kompt berichtung der frag.

9.4 Das Newton–Verfahren und Fixpunktiterationen

Das Sekantenverfahren kann für beliebige stetige Funktionen durchgeführt werden – die zweimalige stetige Differenzierbarkeit brauchen wir "nur", um die Konvergenz und die Konvergenzordnung zu beweisen. Ist $f \in C^1(\mathbb{R})$ (beziehungsweise auf einem Intervall, in dem die Nullstellensuche durchgeführt werden soll), dann kann man die Sekante auch durch die *Tangente* an f in $x^{(k)}$ ersetzen und mit deren Nullstelle weiteriterieren. Das Iterationsverfahren lautet jetzt also

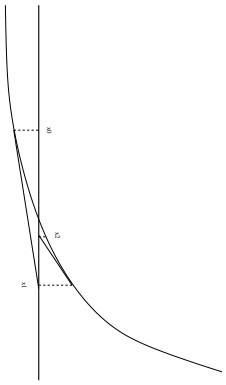


Abbildung 9.4: Geometrische Interpretation des Newton–Verfahrens.

$$x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)}, \qquad k \in \mathbb{N}_0, \qquad x^{(0)} \in \mathbb{R}; \tag{9.10}$$

offensichtlich gibt es Schwierigkeiten an Punkten mit Ableitung 0, also mit horizontaler Tangente. Die ist nicht unerwartet, denn Geraden, die parallel zur x–Achse verlaufen, lassen sich sehr schwer mit selbiger schneiden. Die Konvergenzaussage für das Newton–Verfahren ist sogar noch besser als für das Sekantenverfahren.

Satz 9.7 *Es sei $x \in \mathbb{R}$ eine* einfache *Nullstelle von $f \in C^2(U_x)$ für eine offene Umgebung U_x von x. Dann ist das Newton–Verfahren lokal quadratisch konvergent gegen x.*

Beweis: Die Bestimmung von M_ε ist genau wie im Beweis von Satz 9.6. Über die verallgemeinerte Definition[139] der dividierten Differenzen mit mehrfachen Knoten,

$$[\underbrace{x, \ldots, x}_{n+1}] f := \frac{f^{(n)}(x)}{n!},$$

ist nun ganz analog

$$\begin{aligned}
x^{(k+1)} - x &= x^{(k)} - x - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)} = \left(x^{(k)} - x\right)\left(1 - \frac{f\left(x^{(k)}\right) - f(x)}{\left(x^{(k)} - x\right) f'\left(x^{(k)}\right)}\right) \\
&= \left(x^{(k)} - x\right)\left(1 - \frac{\left[x^{(k)}, x\right] f}{\left[x^{(k)}, x^{(k)}\right] f}\right) \\
&= \frac{\left(x^{(k)} - x\right)^2}{\left[x^{(k)}, x^{(k)}\right] f} \frac{\left[x^{(k)}, x^{(k)}\right] f - \left[x^{(k)}, x\right] f}{\left(x^{(k)} - x\right)} = \left(x^{(k)} - x\right)^2 \frac{\left[x^{(k)}, x^{(k)}, x\right] f}{\left[x^{(k)}, x^{(k)}\right] f} \\
&= \left(x^{(k)} - x\right)^2 \frac{f''(\xi_2)}{2f'(\xi_1)}.
\end{aligned}$$

Der übrige Konvergenzbeweis läuft wieder wie bei Satz 9.6, die Konvergenzordnung ergibt sich aus

$$\lim_{k \to \infty} \frac{\left|x^{(k+1)} - x\right|}{\left|x^{(k)} - x\right|^2} = \left|\frac{f''(x)}{2f'(x)}\right|.$$

□

Das Newton–Verfahren ist eine Iteration der Form

$$x^{(k+1)} = \Phi\left(x^{(k)}\right), \qquad k \in \mathbb{N}_0, \tag{9.11}$$

und das Auffinden einer Nullstelle von f ist nichts anderes als das Auffinden eines *Fixpunkts* von Φ, das heißt, falls $f'(x) \neq 0$, dann ist

$$f(x) = 0 \qquad \Longleftrightarrow \qquad x = \Phi(x) := x - \frac{f(x)}{f'(x)}.$$

Die "Standardaussage" über die Existenz eines Fixpunkts ist der folgende Satz, den wir jetzt, als Gegenstück zu der Version aus Satz 6.2 mal für allgemeine metrische Räume formulieren[140].

[139] Diese Erweiterung ist konsistent! Siehe z.B. [38, Korollar 7.28].

[140] Den Banachschen Fixpunktsatz kann man gar nicht oft genug beweisen und eine schöne Wiederholung ist's allemal.

Satz 9.8 *(Banachscher Fixpunktsatz)*
Sei X ein vollständiger metrischer Raum und $\Phi : X \to X$ eine Kontraktion, das heißt,

$$d\left(\Phi(x), \Phi(y)\right) \leq \gamma\, d(x,y), \qquad 0 < \gamma < 1, \qquad x, y \in X.$$

Dann besitzt Φ einen eindeutig bestimmten Fixpunkt $x^* = \Phi\left(x^*\right)$ und für jeden Startwert $x^{(0)} \in X$ konvergiert die Folge $x^{(k+1)} = \Phi\left(x^{(k)}\right)$, $k \in \mathbb{N}_0$, gegen x^* und es ist

$$d\left(x^*, x^{(k)}\right) \leq \frac{\gamma^k}{1-\gamma}\, d\left(x^{(1)}, x^{(0)}\right). \tag{9.12}$$

Beweis: Beginnen wir mit der Eindeutigkeit. Für zwei Fixpunkte $x^*, y^* \in X$ ist

$$d\left(x^*, y^*\right) = d\left(\Phi\left(x^*\right), \Phi\left(y^*\right)\right) \leq \gamma\, d\left(x^*, y^*\right) \qquad \Longrightarrow \qquad x^* = y^*.$$

Für die Existenz des Fixpunkts bemerken wir zuerst, daß für $k \in \mathbb{N}_0$,

$$d\left(x^{(k+1)}, x^{(k)}\right) = d\left(\Phi\left(x^{(k)}\right), \Phi\left(x^{(k-1)}\right)\right) \leq \gamma\, d\left(x^{(k)}, x^{(k-1)}\right) \leq \cdots \leq \gamma^k\, d\left(x^{(1)}, x^{(0)}\right),$$

und somit, für $m > 0$,

$$d\left(x^{(k+m)}, x^{(k)}\right) \leq \sum_{j=0}^{m-1} d\left(x^{(k+j+1)}, x^{(k+j)}\right) \leq \gamma^k\, d\left(x^{(1)}, x^{(0)}\right) \sum_{j=0}^{m-1} \gamma^j \leq \frac{\gamma^k}{1-\gamma} d\left(x^{(1)}, x^{(0)}\right),$$

also ist $x^{(k)}$, $k \in \mathbb{N}_0$, eine *Cauchyfolge*, die wegen der Vollständigkeit[141] von X gegen einen Grenzwert x^* konvergiert, für den offensichtlich (9.12) gilt. Weil für beliebiges $k \in \mathbb{N}_0$

$$\begin{aligned} d\left(\Phi\left(x^*\right), x^*\right) &\leq d\left(\Phi\left(x^*\right), \Phi\left(x^{(k)}\right)\right) + d\left(x^{(k+1)}, x^{(k)}\right) + d\left(x^*, x^{(k)}\right) \\ &\leq (1+\gamma)\; d\left(x^*, x^{(k)}\right) + \gamma^k\, d\left(x^{(1)}, x^{(0)}\right) \end{aligned}$$

und weil die rechte Seite für $k \to \infty$ gegen 0 konvergiert, ist x^* der gesuchte Fixpunkt. □

Nun können wir ein einfaches Konvergenzkriterium für Fixpunktverfahren mit Funktionen $\Phi : \mathbb{R}^n \to \mathbb{R}^n$ geben. Dafür erinnern wir uns an die Definition des *Spektralradius* einer Matrix $A \in \mathbb{R}^{n\times n}$ als

$$\rho(A) = \max\left\{|\lambda| \;:\; \ker{}_{\mathbb{C}^n}\left(A - \lambda I\right) \neq \{0\}\right\} = \liminf_{k\to\infty} \left\|A^k\right\|^{1/k}. \tag{9.13}$$

siehe z.B. [38, Definition 6.3, Proposition 6.4]. Außerdem bezeichnen wir die *Jacobi–Matrix*[142] einer differenzierbaren Funktion $\Phi : \mathbb{R}^n \to \mathbb{R}^n$ mit $J\left[\Phi\right]$:

$$J\left[\Phi\right] = \left[\frac{\partial \Phi_j}{\partial x_k} \;:\; j,k = 1,\ldots,n\right].$$

[141] Wie man sieht wird in diesem Bereich wirklich jede Voraussetzung gebraucht!

[142] Das ist nun schon die zweite Jacobi–Matrix, die hier auftaucht, aber, Kontext sei Dank, Verwechslungen sollten eigentlich ausgeschlossen sein.

Proposition 9.9 *Ist $\Phi : \mathbb{R}^n \to \mathbb{R}^n$ auf einer offenen Menge $U \subset \mathbb{R}^n$ stetig differenzierbar und gibt es einen Fixpunkt $x \in U$, so daß $x = \Phi(x)$ und $\rho\left(J\left[\Phi\right](x)\right) < 1$, dann ist das Iterationsverfahren $x^{(k+1)} = \Phi\left(x^{(k)}\right)$, $k \in \mathbb{N}_0$, lokal konvergent gegen x.*

Übung 9.3 Zeigen Sie, daß es für jede Matrix $A \in \mathbb{R}^{n\times n}$ und zu jedem $\varepsilon > 0$ eine *Vektornorm* $\|\cdot\|$ gibt, deren zugehörige *Operatornorm* die Bedingung $\|A\| < \rho(A) + \varepsilon$ erfüllt.

Beweis: Ist $\rho\left(J\left[\Phi\right](x)\right) < 1$, dann gibt es[143] eine Vektornorm $\|\cdot\|$, so daß $\|J\left[\Phi\right](x)\| < 1$ und wegen der stetigen Differenzierbarkeit von Φ und der Stetigkeit der Norm gibt es für jedes $\|J\left[\Phi\right](x)\| < \gamma < 1$ ein $\varepsilon > 0$, so daß

$$\sup_{y\in B_\varepsilon(x)} \|J\left[\Phi\right](y)\| < \gamma.$$

Nach dem Hauptsatz der Differential- und Intgralrechnung ist für $y, y' \in B_\varepsilon(x)$

$$\begin{aligned}
\Phi(y) - \Phi\left(y'\right) &= \int_0^1 \frac{d}{dt}\Phi\left(y' + t(y-y')\right)\, dt = \int_0^1 \frac{d}{dt}\Phi\left(ty + (1-t)y'\right)\, dt \\
&= \int_0^1 D_{y-y'}\Phi\left(ty + (1-t)y'\right)\, dt \\
&= \int_0^1 \left(J\left[\Phi\right]\left(ty + (1-t)y'\right)\right)\, \left(y - y'\right)\, dt,
\end{aligned}$$

also

$$\|\Phi(y) - \Phi\left(y'\right)\| \leq \max_{\xi\in B_\varepsilon(x)} \|J\left[\Phi\right](\xi)\|\ \|y - y'\| = \gamma\,\|y - y'\|\,,$$

und damit ist Φ auf $B_\varepsilon(x)$ eine Kontraktion, die nach Satz 9.8 für alle Startwerte $x^{(0)} \in B_\varepsilon(x)$ gegen den eindeutigen Fixpunkt x konvergiert. Nur der Vollständigkeit halber: Daß für $x^{(0)} \in B_\varepsilon(x)$ die Folge $x^{(k)}$ immer in $B_\varepsilon(x)$ bleibt sieht man daran, daß

$$\left\|x^{(k+1)} - x\right\| = \left\|\Phi\left(x^{(k)}\right) - \Phi\left(x\right)\right\| \leq \gamma\left\|x^{(k)} - x\right\| \leq \gamma\,\varepsilon < \varepsilon.$$

□

Bemerkung 9.10 *(Zu Proposition 9.9)*

1. *Der Trick beim Konvergenzbeweis bestand darin, die Norm so geeignet zu wählen, daß wir bezüglich dieser Norm eine Kontraktion erhalten und so den Banachschen Fixpunktsatz anwenden können. Wegen der Äquivalenz* aller *Normen auf dem $\mathbb{R}^n$ konvergiert dann das Newton–Verfahren aber auch für jede beliebige andere Norm.*

[143] Siehe Übungsaufgabe 9.3.

2. *Die Existenz eines Fixpunkts muß angenommen werden und folgt nicht automatisch. Beispielsweise ist die Ableitung der Abbildung* $\Phi(x) = x^2 + 1$ *auf dem Intervall* $\left(-\frac{1}{2}, \frac{1}{2}\right)$ *kleiner als* 1 *und damit ist auch* $|\Phi(x) - \Phi(x)| < |x - y|$ *für* $x, y \in \left(-\frac{1}{2}, \frac{1}{2}\right)$*, aber mit* $x^{(0)} = 0$ *erhalten wir die divergente Iterationsfolge*

$$x^{(1)} = 1, \qquad x^{(2)} = 2, \qquad x^{(3)} = 5, \qquad \dots$$

Die Funktion $\Phi(x) = x^2 - 1$ *hat hingegen zwei Fixpunkte, nämlich* $\frac{1}{2}\left(1 \pm \sqrt{5}\right)$*, aber keiner von beiden liegt im Kontraktivitätsbereich. In diesem Fall erhalten wir sowohl oszillierende Folgen* $x^{(2k)} = 0$*,* $x^{(2k+1)} = -1$*, wie auch divergente: Ist* $x = \frac{1}{2}\left(1 \pm \sqrt{5}\right) + y =: x_1 + y$*,* $y > 0$*, dann ist*

$$\Phi(x) = \underbrace{-1 + x_1^2}_{=x_1} + 2x_1 y + y^2 = x_1 + 2x_1 y + y^2 > x_1 + 2y,$$

und damit divergiert die "Fixpunktiteration" für jeden Startwert "rechts von" x_1 *– von lokaler Konvergenz kann also nicht die Rede sein.*

Man kann aus Fixpunktiterationen auch unter gewissen Voraussetzungen die Konvergenzordnung ablesen: Ist nämlich

$$x = \Phi(x) \qquad \text{und} \qquad D^k \Phi(x) = 0, \quad k = 1, \dots, p-1,$$

aber $\frac{\partial^p}{\partial x^\alpha}\Phi(x) \neq 0$ für mindestens einen Multiindex $\alpha \in \mathbb{N}_0^n$, dann ist nach der Taylor–Formel

$$\begin{aligned} x^{(k+1)} - x &= \Phi\left(x^{(k)}\right) - x \\ &= \underbrace{\sum_{j=1}^{p-1} \sum_{|\alpha|=j} \frac{1}{\alpha!} \frac{\partial^j \Phi}{\partial x^\alpha}(x) \left(x^{(k)} - x\right)^\alpha}_{=0} + \sum_{|\alpha|=p} \frac{1}{\alpha!} \frac{\partial^p \Phi}{\partial x^\alpha}(x) \left(x^{(k)} - x\right)^\alpha + O\left(\left\|x^{(k)} - x\right\|^{p+1}\right), \end{aligned}$$

und wenn $x^{(k)} \to x$ für $k \to \infty$, dann ist

$$\limsup_{k\to\infty} \frac{\left\|x^{(k+1)} - x\right\|_\infty}{\left\|x^{(k)} - x\right\|_\infty^p} \leq \max_{|\alpha|=p} \left|\frac{\partial^p \Phi}{\partial x^\alpha}(x)\right|,$$

was auf Konvergenzordnung $\geq p$ schließen lässt[144]

Mit dieser Theorie können wir jetzt auch das Newton–Verfahren für *Systeme* nichtlinearer Gleichungen, also unser $F(x) = 0$ aus (9.1), in Angriff nehmen. Ist nämlich $F : \mathbb{R}^n \to \mathbb{R}^n$ differenzierbar, so können wir die Iteration

$$x^{(k+1)} = x^{(k)} - J^{-1}[F]\left(x^{(k)}\right) \; F\left(x^{(k)}\right), \qquad k \in \mathbb{N}_0, \qquad x^{(0)} \in \mathbb{R}^n, \tag{9.14}$$

verwenden. Und siehe da – dieses Verfahren konvergiert!

[144] Das ist nicht ganz konsistent mit unserer Definition von Konvergenzordnung, aber erstens machen es alle Bücher so und zweitens können alle die es nicht glauben, gerne die Details selbst ausarbeiten.

Satz 9.11 *Ist $F : \mathbb{R}^n \to \mathbb{R}^n$ zweimal stetig differenzierbar und ist $x \in \mathbb{R}^n$ eine einfache Nullstelle von F, das heißt,*

$$F(x) = 0, \qquad \textit{und} \qquad \det\ J[F](x) \neq 0, \tag{9.15}$$

dann ist das Newton–Verfahren (9.14) lokal konvergent gegen x.

Beweis: Nach Proposition 9.9 müssen wir "nur" zeigen, daß die Iterationsfunktion

$$\Phi(x) := x - J^{-1}\left[F\right](x)\ F(x), \qquad x \in \mathbb{R}^n,$$

die Bedingung $\rho\left(J\left[\Phi\right](x)\right) < 1$ erfüllt. Aus der zweimaligen Differenzierbarkeit von F werden wir sogar mehr bekommen, nämlich

$$J\left[\Phi\right](x) = 0. \tag{9.16}$$

Da F zweimal stetig differenzierbar ist, ist für $y \in \mathbb{R}^n$

$$F(x+y) = \underbrace{F(x)}_{=0} + J[F](x)\ y + \underbrace{\left[\frac{1}{2} y^T \left[\frac{\partial^2 F_\ell}{\partial x_j \partial x_k}\left(\xi_{jk}\right)\right] y\ :\ \ell = 1, \ldots, n\right]}_{=:G(x,y)},$$

wobei $\xi_{j,k} \in [x, x+y]$, $j,k = 1,\ldots,n$, und

$$\lim_{y \to 0} \left[\frac{\partial^2 F_\ell}{\partial x_j \partial x_k}\left(\xi_{jk}\right)\right] = \left[\frac{\partial^2 F_\ell}{\partial x_j \partial x_k}(x)\right] =: H\left[F_\ell\right](x), \qquad \ell = 1, \ldots, n,$$

weil F zweimal stetig differenzierbar ist[145]. Daraus können wir nun folgern, daß es eine Konstante $C > 0$ gibt, so daß

$$\left\|G(x,y)\right\|_\infty \leq C \max_{\ell=1,\ldots,n} \left\| \left[\frac{\partial^2 F_\ell}{\partial x_j \partial x_k}(x)\right] \right\|_\infty \left\|y\right\|_\infty^2 . \tag{9.17}$$

Damit erhalten wir, daß

$$\begin{aligned}
\Phi(x+y) - \Phi(x) &= (x+y) - J^{-1}[F](x+y)\ F(x+y) - x + J^{-1}[F](x)\ F(x) \\
&= y - J^{-1}[F](x)\left(F(x+y) - F(x)\right) + \left(J^{-1}[F](x) - J^{-1}[F](x+y)\right) F(x+y) \\
&= y - J^{-1}[F](x)\left(J[F](x)\ y + G(x,y)\right) \\
&\qquad + \left(J^{-1}[F](x) - J^{-1}[F](x+y)\right) \left(\underbrace{F(x)}_{=0} + J[F](\xi)\ y\right) \\
&= -J^{-1}[F](x)\ G(x,y) + \left(J^{-1}[F](x) - J^{-1}[F](x+y)\right) J[F](\xi)\ y,
\end{aligned}$$

[145] Die $n \times n$–Matrix $H\left[f\right] = \left[\frac{\partial^2 f}{\partial x_j \partial x_k}\ :\ j,k = 1,\ldots,n\right]$ bezeichnet man (bekanntlich ?) als *Hesse–Matrix* der *skalarwertigen* Funktion f.

wobei $\xi \in [x, x+y]$. Die Funktion $J^{-1}[F]$ existiert in einer Umgebung von x und ist dort differenzierbar (weil $J[F]$ differenzierbar ist), also gibt es eine Konstante $D > 0$, so daß

$$\left\| J^{-1}[F](x) - J^{-1}[F](x+y) \right\|_\infty \leq D \left\| y \right\|_\infty ,$$

also ist, für hinreichend kleine y

$$\left\| \left(J^{-1}[F](x) - J^{-1}[F](x+y) \right) J[F](\xi)\, y \right\|_\infty \leq (D + 2\pi)\ \left\| J[F](x) \right\|_\infty\ \left\| y \right\|_\infty^2 . \qquad (9.18)$$

Aus (9.17) und (9.18) folgt nun, daß es eine[146] Konstante $E > 0$ gibt, so daß

$$\left\| \Phi(x+y) - \Phi(x) \right\|_\infty \leq E \left\| y \right\|_\infty^2 \qquad \Longrightarrow \qquad \frac{\left\| \Phi(x+y) - \Phi(x) \right\|_\infty}{\left\| y \right\|_\infty} \leq E \left\| y \right\|_\infty .$$

Dies bedeutet aber nichts anderes als daß Φ an x differenzierbar ist und $J[\Phi](x) = 0$. □

Korollar 9.12 *Das Newton–Verfahren (9.14) konvergiert für zweimal stetig differenzierbare Funktionen F lokal von quadratischer Ordnung gegen eine einfache Nullstelle x von F.*

Bemerkung 9.13 *(Newtonverfahren)*

1. *Die große Schwäche des Newtonverfahrens (und eigentlich aller Fixpunktiterationen) ist die lediglich* lokale *Konvergenz, weswegen die Wahl eines guten Startwerts entscheidend ist. Im Falle eines "falschen" Startwerts kann die Folge divergieren, oszillieren oder auch gegen eine andere Nullstelle konvergieren.*

2. *Die hier vorgestellten Newton–Verfahren sind sehr empfindlich gegenüber* doppelten Nullstellen.

3. *Bei der praktischen Ausführung des Newton–Verfahrens müssen Ableitungen oder, im Fall $n > 1$ eine volle Jacobi–Matrix bestimmt werden. Das ist nicht so schlimm im Falle von Polynomen, deren Ableitungen beim Hornerschema einfach mitberechnet werden können, bei komplizierteren Funktionen hingegen muß man numerisch differenzieren, was den einzelnen Iterationsschritt sehr aufwendig machen kann.*

4. *Beim* Broyden–Verfahren *[7] wird dieses Problem umgangen, indem man aus einer näherungsweisen Jacobi–Matrix B_k an $x^{(k)}$ über die Iteration*

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - \lambda_k\, B_k^{-1} F\left(x^{(k)}\right) \\ B_{k+1} &= B_k + \frac{1}{\left\| x^{(k+1)} - x^{(k)} \right\|_2^2} \left(\Delta F - B_k \Delta x \right) \left(\Delta x \right)^T , \end{aligned}$$

eine neue näherungsweise Jacobi–Matrix B_{k+1} bestimmt, wobei $\Delta F = F\left(x^{(k+1)}\right) - F\left(x^{(k)}\right)$ und $\Delta x = x^{(k+1)} - x^{(k)}$. Die Schrittweite λ_k wird dabei über ein Minimierungsproblem bestimmt, wenn sie aber "zu klein" wird, bleibt auch nichts anderes übrig, als B_{k+1} durch numerische Differentiation zu bestimmen. Mehr Information hierzu findet sich in [47, S. 246–249] und vor allem auch in [26].

[146] letzte ...

5. *Für $n = 1$ gibt es eine einfache Methode zur Konvergenzbeschleunigung von Iterationsverfahren, die auf Aitken zurückgeht. Setzt man $\Delta x^{(k)} := x^{(k+1)} - x^{(k)}$, und konvergiert die Folge $x^{(k)}$ mit Konvergenzordnung*[147] *≥ 1 gegen x, dann konvergiert die Folge*

$$y^{(k)} := x^{(k)} - \frac{\left(\Delta x^{(k)}\right)^2}{\Delta^2 x^{(k)}} = x^{(k)} - \frac{\left(x^{(k+1)} - x^{(k)}\right)^2}{x^{(k+2)} - 2x^{(k+1)} + x^{(k)}}$$

schneller *gegen x, es gilt nämlich*

$$\lim_{k\to\infty} \frac{y^{(k)} - x}{x^{(k)} - x} = 0.$$

Details finden sich beispielsweise in [47, S. 274–279].

9.5 Eine Anwendung – das Bierkastenproblem

Es ist vielleicht einmal ganz illustrativ, zu sehen, wie man von einem realen Anwendungsproblem zum Newtonverfahren und dessen ganz praktischen Schwierigkeiten kommen kann. Das folgende Problem stammt aus der Automatisierungstechnik und beschäftigt sich mit Robotern, die Getränkekisten, genauer Stapel von Getränkekisten[148], bewegt und zwar ber ein Hindernis hinweg. Um dieses Hindernis zu umfahren, muss die Bahn des Roboters durch einen bestimmten Punkt hindurchgeführt werden. Aus technischen Gründen[149] ist die Bahn eine Kombination aus Geradenstücken und Kreisbögen mit *einem* festen Radius r. Damit die Bierkästen

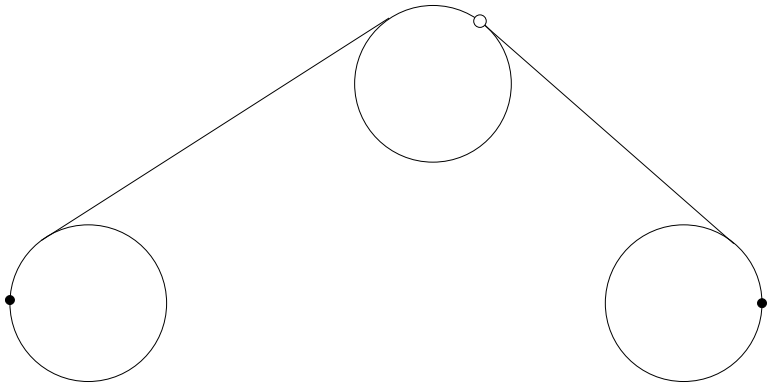


Abbildung 9.5: Eine zulässige Bahn durch den vorgegebenen Punkt. Die Kreise mit dem vorgegebeben Radius sind eingezeichnet.

nicht umfallen muss die Bahn ausserdem im Anfangs- und Endpunkt eine senkrechte Tangente haben, siehe Abb. 9.5. Die Aufgabe ist nun:

[147] Achtung: Das bedeutet bereits *exponentielle* Konvergenz.
[148] Daher der interne "Codename" des Projekts: *Das Bierkastenproblem*
[149] So ist das Ding halt einfach gebaut.

Wie bestimmt man den kürzesten derartigen Weg.

Dabei überlegt man sich zunächst, daß sich das Problem ziemlich vereinfachen lässt. Da der Pfad senkrecht beginnt und endet, besteht er immer aus einem Halbkreis und zwei Geraden, siehe Abb. 9.6. Die beiden Mittelpunkte sind durch Anfangs- bzw. Endpunkt und Radius ein-

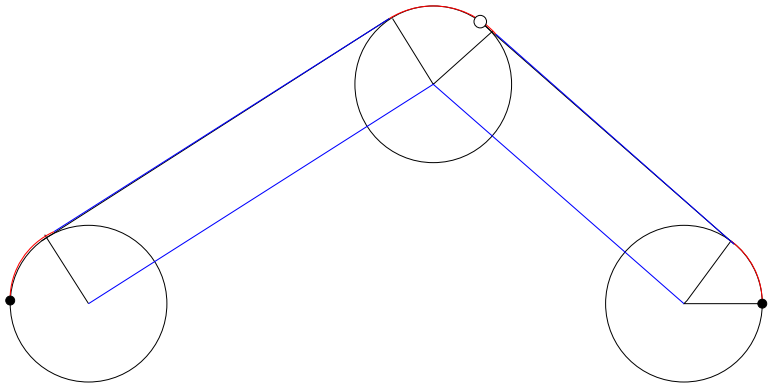


Abbildung 9.6: Die Kreis- und Geradenanteile. Genau genommen interessiert einen also nur ein Pfad vom ersten Kreismittelpunkt über einem Punkt, dessen Abstand zum "Ausweichpunkt" mit dem Radius übereinstimmt zum Mittelpunkt des zweiten Kreises.

deutig bestimmt, der einzige "Freiheitsgrad" besteht also in der Wahl des Punktes, an dem sich die beiden Linienstücke treffen und dieser muss auf einem Kreis mit dem Radius r um den Ausweichpunkt c liegen, siehe Abb. 9.7, und zwar so, daß der **Gesamtweg** minimal wird. Das

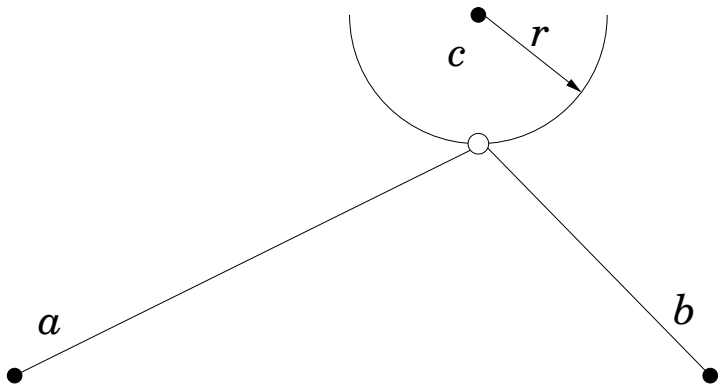


Abbildung 9.7: Das wirkliche Optimierungsproblem.

Optimierungsproblem besteht also darin, einen Punkt x auf dem Kreis so zu wählen, daß der Gesamtweg minimiert wird, also

$$\min_x w(x) := \min_x \|x-a\|_2 + \|x-b\|_2\,, \qquad \|x-c\|_2 = r. \tag{9.19}$$

Das Gemeine an der Sache ist, daß man hier wirklich w, also die Summe der beiden Streckenlängen zu minimieren hat und nicht den sehr populären Wert $\|x-a\|_2^2+\|x-b\|_2^2$. Durch die unscheinbare Quadrierung wird das Problem deutlich einfacher, allerdings ist halt dann auch die Lösung eine andere.

Wir bemerken zuerst einmal, daß das Minimum eindeutig sein muss, es kann also nur genau einen Punkt geben, für den der Weg am kürzesten ist. Das ist eine Konsequenz aus der (strikten) Konvexität der Norm, die besagt, daß

$$\left\|\frac{x+x'}{2}\right\|_2 < \frac{1}{2}\left(\|x\|_2+\|x'\|_2\right)$$

ist. Wären nun x, x' zwei Punkte auf dem Kreis, für die der Weg $w(x) = w\left(x'\right)$ minimal ist, dann ist

$$\begin{aligned} w\left(\frac{x+x'}{2}\right) &= \left\|\frac{x+x'}{2}-a\right\|_2+\left\|\frac{x+x'}{2}-b\right\|_2 \\ &= \left\|\frac{1}{2}\left[(x-a)+(x'-a)\right]\right\|_2+\left\|\frac{1}{2}\left[(x-b)+(x'-b)\right]\right\|_2 \\ &< \frac{1}{2}\left(\|x-a\|_2+\|x'-a\|_2+\|x-b\|_2+\|x'-b\|_2\right)=\frac{1}{2}\left(w(x)+w(x')\right), \end{aligned}$$

und der Weg über den Mittelpunkt wäre kürzer. Nun liegt der Mittelpunkt aber **im** Kreis, doch wenn wir ihn mit dem Kreiszentrum verbinden und diese Gerade wieder mit dem Kreis schneiden, dann erhalten wir sogar einen noch kürzeren Weg.

Mit demselben Argument erhalten wir auch, daß $w(x)$ keine lokalen Minima auf dem Kreis haben, also "auf und ab" gehen kann. Gäbe es einen Punkt x, so daß zwischen x und der Minimalstelle x^* größere Werte als $w(x)$ auftauchen, dann gibt es einen "nächstgelegenen" Punkt x', so daß $w(x) = w(x') \leq w(y)$ für alle $y \in [x, x']$ ist, was aber für den Mittelpunkt $\frac{1}{2}\left(x+x'\right)$ nicht zutrifft. Mit anderen Worten: $w(x)$ fällt monoton bis $w\left(x^*\right)$ und steigt von dort aus wieder.

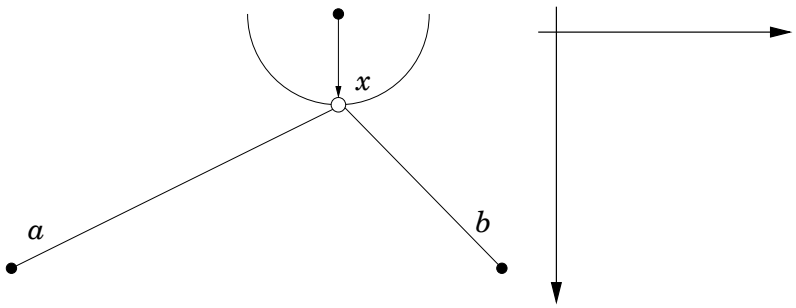


Abbildung 9.8: Parametrisierung und Koordinatensystem.

Nun parametrisieren wir x als

$$x = c + r\begin{bmatrix}\sin\phi\\ \cos\phi\end{bmatrix}, \qquad \phi \in [-\pi, \pi],$$

nehmen der Einfachheit an, daß $r = 1$ ist[150], und erhalten, daß

$$\begin{aligned} w(x) &= w(\phi) = \ell_a(\phi) + \ell_b(\phi) \\ &:= \sqrt{(c_1 + \sin\phi - a_1)^2 + (c_2 + \cos\phi - a_2)^2} \\ &\quad + \sqrt{(c_1 + \sin\phi - b_1)^2 + (c_2 + \cos\phi - b_2)^2}. \end{aligned}$$

Dann ist

$$\begin{aligned} \ell_a'(\phi) &= \frac{2\,(c_1 - a_1 + \sin\phi)\cos\phi - 2\,(c_2 - a_2 + \cos\phi)\sin\phi}{2\ell_a(\phi)} \\ &= \frac{(c_1 - a_1)\cos\phi - (c_2 - a_2)\sin\phi}{\ell_a(\phi)} \end{aligned}$$

und somit

$$w'(\phi) = \frac{(c-a)^T\, s'(\phi)}{\ell_a(\phi)} + \frac{(c-b)^T\, s'(\phi)}{\ell_b(\phi)}, \qquad s(\phi) = \begin{bmatrix} \sin\phi \\ \cos\phi \end{bmatrix}.$$

Auf der anderen Seite ist aber auch

$$\begin{aligned} \ell_a^2(\phi) &= (c_1 + \sin\phi - a_1)^2 + (c_2 + \cos\phi - a_2)^2 \\ &= (c_1 - a_1)^2 + 2\,(c_1 - a_1)\sin\phi + \sin^2\phi + (c_2 - a_2)^2 + 2\,(c_2 - a_2)\cos\phi + \cos^2\phi \\ &= 1 + \|c-a\|_2^2 + 2\,(c-a)^T\, s\,(\phi) =: 1 + \|c-a\|_2^2 + 2\,f_a(\phi), \end{aligned}$$

also

$$w'(\phi) = \frac{f_a'(\phi)}{\sqrt{1 + \|c-a\|_2^2 + 2\,f_a(\phi)}} + \frac{f_b'(\phi)}{\sqrt{1 + \|c-b\|_2^2 + 2\,f_b(\phi)}}.$$

Der Nenner dieses Ausdrucks ist $\neq 0$ wenn $\|c-a\| > 1$ und $\|c-b\| > 1$ sind, was auch geometrisch–anschaulich eine vernünftige Annahme darstellt: Anfangs- und Endpunkt liegen ausserhalb des Kreises um c.

Damit könne wir unsere Bestimmungsgleichung $0 = w'(\phi)$ für das Minimum in

$$0 = g(\phi) = \ell_b(\phi)\, f_a'(\phi) + \ell_a(\phi)\, f_b'(\phi)$$

umformen. Da

$$g'(\phi) = \ell_b'(\phi)\, f_a'(\phi) + \ell_b(\phi)\, f_a''(\phi) + \ell_a'(\phi)\, f_b'(\phi) + \ell_a(\phi)\, f_b''(\phi)$$

ist, können wir nun (versuchen,) ϕ über eine univariate Newton–Iteration oder über Bisektion zu bestimmen.

Betrachtet man die Menge der Punkte x, für wie $w(x)$ konstant ist, also die Niveaulinien von w, dann ist dies gerade eine Ellipse mit den beiden Brennpunkten a und b. Diese Ellipse hat die beiden Halbachsen α und β, die die Bedingung $\alpha^2 + \beta^2 = \|a-b\|^2$ erfüllen. Jeder Punkt auf der Ellipse hat dann in der Summe den Abstand 2α von den beiden Brennpunkten und die

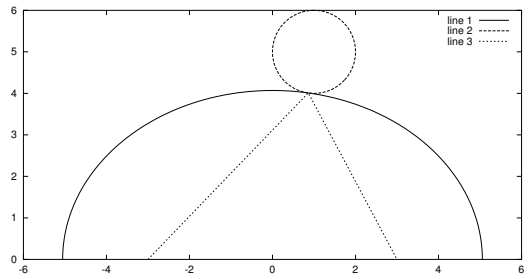


Abbildung 9.9: Berührellipse an die Kugel und der zugehörige kürzeste Weg.

Suche nach dem kürzesten Weg ist äquivalent zum Auffinden der Berührellipse mit der großen Halbachse α, siehe Abb. 9.9. Dazu empfiehlt es sich, ein paar Vereinfachungen zu machen. Wir setzen $e = \|a - b\|_2$ und wählen $a = [-e, 0]$, $b = [0, e]$ und legen damit den Ursprung in die Mitte zwischen den beiden Punkten. Die Gleichung für die Ellipse ist dann

$$\frac{x^2}{\alpha^2} + \frac{y^2}{\beta^2} = 1, \qquad \alpha^2 - \beta^2 = e^2 = \|a - b\|_2^2, \tag{9.20}$$

und die der Kugel ja bekanntlich

$$(x - c_1)^2 + (y - c_2)^2 = r^2. \tag{9.21}$$

Dieses Gleichungssystem aus (9.20) und (9.21) hat im allgemeinen vier Lösungen, beispielsweise, wenn $c = 0$ und $\beta < r < \alpha$ ist. Zwei der Lösungen sind anschaulich sofort klar, nämlich der nachste Punkt wie in Abb 9.9 und die Berührellipse an den fernsten Punkt des Kreises. Die beiden anderen Lösungen entsprechen Ellipsen mit *negativer* Halbachse, deren "Mittelpunkt" im Unendlichen liegt, was zu Berührhyperbeln an den Kreis führt. Das wird sich auch in unserem Fall nicht ändern, und wir müssten α so bestimmen, daß es zwei komplexe und eine doppelte reelle Lösung des Gleichungssystems gibt. Dieses Problem lässt sich mit Methoden der Computeralgebra angehen, ist da aber dann schon ein echter Härtetest, siehe [39].

Um das Problem ausgehend von dieser geometrischen Überlegung numerisch anzugehen, verwenden wir die parametrischen Darstellungen von Kugel und Ellipse als

$$K(\phi) = c - r \begin{bmatrix} \sin\phi \\ \cos\phi \end{bmatrix}, \qquad E(\psi) = \begin{bmatrix} \alpha \sin\psi \\ \beta \cos\psi \end{bmatrix}, \qquad \phi, \psi \in [-\pi, \pi].$$

Wir müssen also α, ϕ und ψ so bestimmen, daß wir einen Berührpunkt erhalten, daß also die Tangenten

$$K'(\phi) = -r \begin{bmatrix} \cos\phi \\ -\sin\phi \end{bmatrix}, \qquad E'(\phi) = \begin{bmatrix} \alpha \cos\psi \\ -\beta \sin\psi \end{bmatrix}$$

[150] So viel Normalisierung kann und darf schon sein.

kollinear sind: $K'(\phi) = \lambda E'(\psi)$, $\lambda \neq 0$. Da die Tangentialvektoren nie $= 0$ sind, können wir ohne weiteres $\lambda \neq 0$ fordern. Setzen wir $s_\phi := \sin\phi$, $c_\phi := \cos\phi$ und dasselbe für ψ, dann erhalten wir das Gleichungssystem

$$\begin{aligned} 0 &= c_1 - r\,s_\phi - \alpha\,s_\psi, \\ 0 &= c_2 - r\,c_\phi - \beta\,c_\psi, \\ 0 &= \lambda\,\alpha\,c_\psi - r\,c_\phi, \\ 0 &= \lambda\,\beta\,s_\psi - r\,s_\phi, \end{aligned}$$

zusammen mit den Nebebedingungen

$$\begin{aligned} 0 &= s_\phi^2 + c_\phi^2 - 1, \\ 0 &= s_\psi^2 + c_\psi^2 - 1, \\ 0 &= \alpha^2 - \beta^2 - e^2. \end{aligned}$$

Und das sind nun wieder sieben Gleichungen in den sieben Unbekannten s_ϕ, c_ϕ, s_ψ, c_ψ, α, β, λ, die sich sehr einfach und schnell mit dem Newton–Verfahren lösen lassen (etwa fünf bis sechs Iterationen für die optimale Genauigkeit von 10^{-16}). Insbesondere ist die Jacobi–Matrix sehr einfach und kommt ohne komplizierte oder numerisch ungenaue Operationen aus.

Ein wenig trickreich ist hierbei allerdings die Wahl des Startwerts. Eine vernünftige erste Näherung ist der Schnittpunkt der Verbindung zwischen c und dem Mittelpunkt $(0,0)$ der Ellipse mit dem Kreis. Dieser Verbindungsvektor ist gerade $0 - c = -c$ und der Schnittpunkt

$$y = c - \frac{r}{\|c\|_2}c =: c - r\begin{bmatrix} s_\phi \\ c_\phi \end{bmatrix} \qquad \Rightarrow \qquad \begin{bmatrix} s_\phi \\ c_\phi \end{bmatrix} = \frac{c}{\|c\|_2}.$$

Der zurückgelegte Weg $w(y)$ ist

$$\|y-(0,-e)\|_2 + \|y-(0,e)\|_2 = 2\alpha, \quad \Rightarrow \quad \alpha = \frac{\|y-(0,-e)\|_2 + \|y-(0,e)\|_2}{2},$$

woraus wir $\beta = \sqrt{\alpha^2 - e^2}$ und die Näherungen für

$$\widetilde{s}_\psi = \frac{c_1 - r s_\phi}{\alpha}, \qquad \widetilde{c}_\psi = \frac{c_2 - r c_\phi}{\beta}$$

erhalten. Das können wir noch zu

$$s_\psi = \frac{\widetilde{s}_\psi}{\widetilde{s}_\psi^2 + \widetilde{c}_\psi^2} \qquad \text{und} \qquad c_\psi = \frac{\widetilde{c}_\psi}{\widetilde{s}_\psi^2 + \widetilde{c}_\psi^2}$$

normieren und haben so den Winkel in der Ellipse angenähert. Nachdem die Tangenten gegenläufig sind, ist $\lambda = -1$ ein ganz gut geratener Wert. Das Ganze funktioniert gut, solange c zwischen den Brennpunkten liegt, genauer, wenn sich die Projektion von c auf die Verbindungsgerade der Brennpunkte zwischen den Brennpunkten befindet. Ansonsten divergiert gelegentlich das Verfahren oder konvergiert gegen die andere Extremlösung, nämlich den Kreispunkt, der zum *längsten* Weg gehört.

9.6 Nullstellen von Polynomen

Eine Schwierigkeit des Newton–Verfahrens, die wir bisher noch gar nicht so recht in Betracht gezogen haben, ist die Bestimmung der Ableitung. Wenn man diese nicht gerade explizit kennt, wie beispielsweise bei Sinus und Cosinus, ist es normalerweise schwierig und aufwendig, sie zu bestimmen – glücklicherweise mit einer Ausnahme, das sind die Polynome.

Dazu erinnern wir uns an das Hornerschema zur Auswertung eines Polynoms

$$p(x) = \sum_{j=0}^{n} p_j x^j, \qquad p_n \neq 0, \qquad x \in \mathbb{R},$$

an der Stelle $\xi \in \mathbb{R}$. Die Berechnungsregel lautet

$$q_n = p_n, \qquad q_j = q_j(\xi) = \xi\, q_{j+1} + p_j, \qquad j = n-1, \ldots, 0 \tag{9.22}$$

wobei dann $p(\xi) = q_0(\xi)$.

Lemma 9.14 *Für $\xi \in \mathbb{R}$ erfüllt das Polynom*

$$q(x) := \sum_{j=0}^{n-1} q_{j+1}(\xi)\; x^j, \qquad x \in \mathbb{R},$$

die Bedingung

$$p(x) = q_0(\xi) + (x-\xi)q(x), \qquad x \in \mathbb{R}. \tag{9.23}$$

Beweis: Iteration von (9.22) liefert, daß

$$q_j = \sum_{k=0}^{n-j} p_{j+k}\xi^k, \qquad j = 0, \ldots, n,$$

also ist

$$(x-\xi)\, q(x) + q_0 = (x-\xi)\sum_{j=0}^{n-1} q_{j+1}x^j + q_0 = \underbrace{q_0 + \sum_{j=0}^{n-1} q_{j+1}x^{j+1}}_{=\sum_{j=0}^{n} q_j x^j} - \sum_{j=0}^{n-1} \xi q_{j+1}(\xi)\; x^j,$$

und

$$\begin{aligned}\sum_{j=0}^{n-1} \xi q_{j+1}(\xi)\; x^j &= \sum_{j=0}^{n-1}\sum_{k=0}^{n-j-1} p_{j+1+k}\xi^{k+1}\; x^j = \sum_{j=0}^{n-1}\sum_{k=1}^{n-j} p_{j+k}\xi^k\; x^j \\ &= \sum_{j=0}^{n}\underbrace{\sum_{k=0}^{n-j} p_{j+k}\xi^k}_{=q_j}\; x^j - \sum_{j=0}^{n} p_j x^j = \sum_{j=0}^{n} q_j x^j - p(x).\end{aligned}$$

□

```
%#
%# Horner2Z.m
%# Hornerschema mit Ableitungsberechnung
%# Daten:
%#   p  Koeffizientenvektor
%#   x  Punkt

function [q,r] = Horner2Z( p,x )
  n = length(p);

  q = r = p(n);

  for j=n-1:-1:2
    q = x*q + p(j);
    r = x*r + q;
  end
  q = x*q + p(1);
%endfunction
```

Programm 9.2 `Horner2Z.m`: Das zweizeilige Hornerschema.

Korollar 9.15 *Für $\xi \in \mathbb{R}$ ist*

$$p'(\xi) = q(\xi). \tag{9.24}$$

Beweis: Ableiten von (9.23) nach x ergibt

$$p'(x) = q(x) + (x - \xi)\ q'(x),$$

was mit $x = \xi$ sofort (9.24) liefert. □

Um q auszuwerten kann man nun wieder auf das Hornerschema zurückgreifen und die gerade berechneten Werte $q_j(\xi)$ verwenden. Das führt zum *zweizeiligen Hornerschema*

$$r_n = q_n = p_n, \qquad q_j = \xi q_{j+1} + p_j, \qquad r_j = \xi r_{j+1} + q_j, \qquad j = n-1, \ldots, 0. \tag{9.25}$$

Korollar 9.16 *Für ein Polynom $p \in \Pi_n$ und $\xi \in \mathbb{R}$ berechnet das zweizeilige Hornerschema (9.25) die Werte $p(\xi) = q_0$ und $p'(\xi) = r_1$.*

Unter Verwendung des zweizeiligen Hornerschemas können wir nun sehr einfach das Newton–Verfahren für Polynome programmieren und so die Nullstellen von Polynomen bestimmen. Unter gewissen Voraussetzungen kann man sogar “extreme” Nullstellen finden.

```
%#
%# PolyNewton.m
%# Nullstelle eines Polynoms mit dem Newton-Verfahren
%# Daten:
%#   p  Koeffizientenvektor
%#   x0 Startwert
%#   t  Toleranz

function x = PolyNewton( p,x0,t )
  x = x0; x0 = 1-x0;
  [f,ff] = Horner2Z( p,x );

  while ( abs(f) > t ) && ( abs(x-x0) > t )
    x0 = x;
    x = x - f/ff;
    [f,ff] = Horner2Z( p,x );
  end
%endfunction
```

Programm 9.3 `PolyNewton.m`: Nullstellen von Polynomen mit dem Newton–Verfahren.

Proposition 9.17 *Sei $p \in \Pi_n$ ein Polynom mit n reellen Nullstellen $\xi_1 \le \cdots \le \xi_n$. Dann konvergiert für alle Startwerte $x^{(0)} \ge \xi_n$ bzw. $x^{(0)} \le \xi_1$ das Newton–Verfahren gegen ξ_n bzw. ξ_1.*

Beweis: Wir bemerken zuerst, daß nach dem Satz von Rolle alle Nullstellen von $p', p'', \ldots$, zwischen ξ_1 und ξ_n liegen. Da p keine weiteren Nullstellen hat, ist entweder $p(x) > 0$, $x > \xi_n$, oder $p(x) < 0$, $x > \xi_n$. Nehmen wir den ersten Fall an[151]. Da, für $x > \xi_n$,

$$0 < p(x) = \underbrace{p(\xi_n)}_{=0} + p'(\xi)\,(x - \xi_n)\,, \qquad \xi \in [\xi_n, x]\,,$$

und da p' rechts von ξ_n keine Nullstelle mehr hat, ist also $p'(x) > 0$, $x > \xi_n$. Sei nun $\xi' \le \xi_n$ die größte Nullstelle von p', dann ist für $x > \xi_n \ge \xi'$ auch

$$0 < p'(x) = \underbrace{p'(\xi')}_{=0} + p''(\xi)\,(x - \xi')\,, \qquad \xi \in [\xi', x]\,,$$

das heißt, $p''(x) > 0$, $x > \xi_n$, und mit Iteration dieses Arguments erhält man, daß

$$\operatorname{sgn} p^{(j)}(x) = \operatorname{sgn} p(x), \qquad x > \xi_n, \quad j = 1, \ldots, n.$$

[151] Das ist keine Einschränkung, ansonsten ersetzen wir p durch $-p$.

Insbesondere ist p rechts von ξ_n konvex und damit ist

$$x^{(k+1)} = x^{(k)} - \underbrace{\frac{p\left(x^{(k)}\right)}{p'\left(x^{(k)}\right)}}_{>0} < x^{(k)},$$

solange $x^{(k)} > \xi_n$. Bleibt zu zeigen, daß auch $x^{(k+1)} > \xi_n$. Dazu bemerken wir, daß

$$\begin{aligned} 0 &= p(\xi_n) = p\left(x^{(k)}\right) + \left(\xi_n - x^{(k)}\right) p'\left(x^{(k)}\right) + \underbrace{\frac{\left(\xi_n - x^{(k)}\right)^2}{2} p'(\xi)}_{>0} \\ &> p\left(x^{(k)}\right) + \left(\xi_n - x^{(k)}\right) p'\left(x^{(k)}\right) \end{aligned}$$

und, nach der Iterationsregel, $p\left(x^{(k)}\right) = \left(x^{(k)} - x^{(k+1)}\right) p'\left(x^{(k)}\right)$, also

$$0 > \left(x^{(k)} - x^{(k+1)} + \xi_n - x^{(k)}\right) p'\left(x^{(k)}\right) = \left(\xi_n - x^{(k+1)}\right) \underbrace{p'\left(x^{(k)}\right)}_{>0},$$

und damit $x^{(k+1)} > \xi_n$. Da die Folge $x^{(k)}$ monoton fallend und $\geq \xi_n$ ist, muß sie einen Grenzwert x^* besitzen und da

$$\left|\frac{p\left(x^{(k)}\right)}{p'\left(x^{(k)}\right)}\right| = \left|x^{(k+1)} - x^{(k)}\right| \to 0$$

konvergiert, muß $p(x^*) = 0$ und damit $x^* = \xi_n$ sein. □

Damit haben wir eine "Strategie", *alle* Nullstellen eines Polynoms zu finden: Man startet das Newton–Verfahren nach Möglichkeit "weit genug" rechts oder links, findet eine Nullstelle[152], dividiert diese ab und sucht weiter, bis man schließlich alle Nullstellen gefunden hat. Das Abdividieren von Nullstellen ist nicht weiter schwer: Für $\xi \in \mathbb{R}$ ist

$$\sum_{j=0}^{n} a_j\, x^j = a_n\, x^{n-1}\,(x-\xi) + \underbrace{(a_{n-1} + a_n \xi)}_{=:b_{n-1}}\, x^{n-1} + \sum_{j=0}^{n-2} \underbrace{a_j}_{=:b_j}\, x^j = a_n\, x^{n-1}\,(x-\xi) + \sum_{j=0}^{n-1} b_j\, x^j;$$

die zugehörige Funktion ist in `DivideZero.m` implementiert. Um nun *alle* Nullstellen zu finden, dividiert man eine gefundene Nullstelle ab und verwendet sie gleichzeitig[153] als Startwert für das nächste Newton–Verfahren.

Beispiel 9.18 *Leider ist die hier vorgestellte Variante des Newton–Verfahrens nicht übermäßig stabil, insbesondere das Abdividieren führt zu Problemen.*

1. *Betrachten wir das Polynom*

$$p(x) = \prod_{j=0}^{13} \left(x - 2^{-j}\right), \qquad x \in \mathbb{R},$$

siehe [47, S. 259–260]. Startet man das Newton–Verfahren mit der größten Nullstelle $x^{(0)} = 1$, dann erhält man die folgenden Nullstellen[154]

[152] Am besten natürlich die betragsgrößte oder betragskleinste.

[153] In der Hoffnung, daß sie eine extremale Nullstelle war.

[154] "$\to$" bedeutet, daß die Rechnung nicht mehr terminiert, also das Verfahren sich "aufhängt".

```
%#
%# DivideZero.m
%# Abdividieren einer Nullstelle, gibt gemachten Fehler als zweiten Wert
%# zurueck
%# Daten:
%#   p  Koeffizientenvektor
%#   x  Nullstelle


function [q,err] = DivideZero( p,x )
  n = length( p );
  q = zeros( 1,n-1 );

  for j = n:-1:2
    t = q( j-1 ) = p( j );
    p( j-1 ) = p( j-1 ) + t*x;
  end
  err = p( 1 );
%endfunction
```

Programm 9.4 `DivideZero.m`: Abdividieren einer Nullstelle.

```
%#
%# PolyZeros.m
%# Finde Nullstellen eines Polynoms mit vorgegebener Toleranz und durch
%# Abdividieren
%# Daten:
%#   p  Koeffizientenvektor
%#   x0 Startwert
%#   t  Toleranz

function z = PolyZeros( p,x0,t )
  n = length( p );
  z = zeros( 1,n-1 );
  x = x0;

  for j = 1:n-1
    z( j ) = x = PolyNewton( p,x,t );
    disp(x);
    p = DivideZero( p,x );
  end
%endfunction
```

Programm 9.5 `PolyZeros.m`: Newton–Verfahren mit Adividieren.

	$\varepsilon = 10^{-6}$		$\varepsilon = \hat{u}$	
Exakt	*Berechnet*	*rel. Fehler*	*Berechnet*	*rel. Fehler*
1	1	0	1	0
0.5	0.50580	0.011600	0.5000	2.6645×10^{-15}
0.25	-0.13001	1.5201	0.25000	2.1575×10^{-11}
0.125	-0.24151	2.9321	0.12500	1.1627×10^{-6}
0.0625	$\rightarrow$		0.062352	0.0023729
0.03125			0.039506	0.26421
0.015625			$\rightarrow$	

der Startwert $x^{(0)} = 2$ *liefert sogar das noch weniger begeisternde Ergebnis*

	$\varepsilon = 10^{-6}$		$\varepsilon = \hat{u}$	
Exakt	*Berechnet*	*rel. Fehler*	*Berechnet*	*rel. Fehler*
1	1.0000	8.5183×10^{-10}	1.0000	2.2204×10^{-16}
0.5	0.50580	0.011592	0.50000	2.4125×10^{-12}
0.25	-0.12996	1.5199	0.25000	1.3175×10^{-8}
0.125	-0.24163	2.9330	0.12500	3.1917×10^{-5}
0.0625	$\rightarrow$		0.058106	0.070300
0.03125			0.054892	0.75654
0.015625			$\rightarrow$	

2. *Ein anderes "Monster" ist das schon klassische* Wilkinson–Polynom

$$p(x) = (x-1)\cdots(x-20)\,, \qquad x \in \mathbb{R},$$

mit den ganz sauber getrennten Nullstellen $1, \ldots, 20$*. Dieses Polynom hat die Rundungsfehleranalyse sehr stark motiviert und beeinflußt, siehe [53]. Die Details sind in [50, 51] ausgearbeitet. Die Koeffizienten dieses Polynoms sind sehr groß (bis zur Größenordnung* 10^{18}*), was relative Fehler unvermeidlich macht, und das Problem ist* extrem *schlecht konditioniert bezüglich dieser Nullstellen. Deswegen ist eine numerische Berechnung praktisch unmöglich.*

Uns ist in alten mæren
wunders viel geseit
von Helden lobebæren
von grôzer arebeit

Das Nibelungenlied

9 Literatur

[1] A. G. Aitken, *On interpolation by iteration of proportional parts, without the use of differences*, Proc. Edinburgh Math. Soc. **3** (1932), 56–76.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK user's guide*, second ed., SIAM, 1995.

[3] S. N. Bernstein, *Démonstration du théorème de Weierstrass, fondée su le calcul des probabilitiés*, Commun. Soc. Math. Kharkov **13** (1912), 1–2.

[4] C. de Boor, *On calculating with B–splines*, J. Approx. Theory **6** (1972), 50–62.

[5] C. de Boor, *A practical guide to splines*, Springer–Verlag, New York, 1978.

[6] C. de Boor, *Splinefunktionen*, Lectures in Mathematics, ETH Zürich, Birkhäuser, 1990.

[7] C. G. Broyden, *A class of methods for solving nonlinear simulaneous equations*, Math. Comp. **19** (1965), 577–593.

[8] P. L. Butzer, M. Schmidt, and E. L. Stark, *Observations on the history of central B–splines*, Archive for History of Exact Sciences **39** (1988), no. 2, 137–156.

[9] P. de Casteljau, *Formes à pôles*, Hermes, Paris, 1985.

[10] C. K. Chui, *Multivariate splines*, CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 54, SIAM, 1988.

[11] M. G. Cox, *The numerical evaluation of B–splines*, J. Inst. Math. Appl. **10** (1972), 134–149.

[12] H. B. Curry and I. J. Schoenberg, *On Pólya frequency functions IV: The fundamental spline functions and their limits*, J. d'Analyse Math. **17** (1966), 71–107.

[13] P. J. Davis, *Interpolation and approximation*, Dover Books on Advanced Mathematics, Dover Publications, 1975.

[14] R. T. Farouki and T. N. T. Goodman, *On the optimal stability of the bernstein basis*, Math. Comp. **65** (1996), 1553–1566.

[15] R. T. Farouki and V. T. Rajan, *On the numerical condition of polynomials in Bernstein form*, Comput. Aided Geom. Design **4** (1987), 191–216.

[16] W. Gautschi, *Numerical analysis. an introduction*, Birkhäuser, 1997.

[17] D. Goldberg, *What every computer scientist should know about floating–point arithmetic*, ACM Computing Surveys **23** (1991), 5–48.

[18] G. Golub and C. F. van Loan, *Matrix computations*, The Johns Hopkins University Press, 1983.

[19] G. H. Golub (ed.), *Studies in numerical analysis*, MAA Studies in Mathematics, vol. 24, The Mathematical Association of America, 1984.

[20] N. J. Higham, *Accuracy and stability of numerical algorithms*, SIAM, 1996.

[21] E. Isaacson and H. B. Keller, *Analysis of Numerical Methods*, John Wiley & Sons, 1966.

[22] D. Jörgensen, *Der Rechenmeister*, Rütten & Loenig, 1999, Roman.

[23] S. Karlin, *Total positivity*, Stanford University Press, Stanford CA, 1968.

[24] K. S. Kunz, *Numerical Analysis*, McGraw-Hill Book Company, 1957.

[25] C. A. Micchelli, *Mathematical aspects of geometric modeling*, CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 65, SIAM, 1995.

[26] J. J. Moré and D. J. Sorensen, *Newton's method*, Studies in Numerical Analysis (G. H. Golub, ed.), MAA Studies in Mathematics, vol. 24, The Mathematical Association of America, 1984, pp. 29–82.

[27] E. H. Neville, *Iterative interpolation*, J. Indian Math. Soc. **20** (1934), 87–120.

[28] G. Nürnberger, *Approximation by spline functions*, Springer–Verlag, 1989.

[29] J. Oates, *Babylon*, Thames and Hudson, 1979, Deutsche Ausgabe: Gustav Lübbe Verlag, 1983. Lizenzausgabe für Gondrom Verlag, 1990.

[30] J. M. Peña (ed.), *Shape preserving representations in Computer Aided Geometric Design*, Nova Science Publishers, 1999.

[31] J. M. Peña and T. Sauer, *On the multivariate Horner scheme*, SIAM J. Numer. Anal. **37** (2000), 1186–1197.

[32] _____, *On the multivariate Horner scheme II: Running error analysis*, Computing **65** (2000), 313–322.

[33] W. Popp, *Wege des exakten denkens. vier jahrtausende mathematik*, Weltbild Verlag, 1987, Originalausgabe Franz Ehrenwirth Verlag, 1981.

[34] L. Ramshaw, *Blossoming: A connect–the-dots approach to splines*, Tech. report, Digital Systems Research Center, Palo Alto, 1987.

[35] _____, *Blossoms are polar forms*, Comp. Aided Geom. Design **6** (1989), 352–378.

[36] A. Riese, *Rechenbuch / auff Linien und Ziphren / in allerley Handhierung / Geschäfften unnd Kauffmannschafft*, Franck. Bey. Chr. Egen. Erben, 1574, Facsimile: Verlag Th. Schäfer, Hannover, 1987.

[37] T. Sauer, *Ein algorithmischer Zugang zu Polynomen und Splines*, Mathem. Semesterberichte **43** (1996), no. , 169–189, Vortrag im *Eichstätter Kolloquium zur Didaktik der Mathematik*.

[38] _____, *Numerische Mathematik I*, Vorlesungsskript, Friedrich–Alexander–Universität Erlangen–Nürnberg, Justus–Liebig–Universität Gießen, 2000, `http://www.math.uni-giessen.de/tomas.sauer`.

[39] _____, *Computeralgebra*, Vorlesungsskript, Justus–Liebig–Universität Gießen, 2001, `http://www.math.uni-giessen.de/tomas.sauer`.

[40] I. J. Schoenberg, *Contributions to the problem of approximation of equidistant data by analytic functions. part A. – on the problem of of smoothing or graduation. a first class of analytic approximation formulae*, Quart. Appl. Math. **4** (1949), 45–99.

[41] _____, *Contributions to the problem of approximation of equidistant data by analytic functions. part B. – on the second problem of osculatory interpolation. a second class of analytic approximation formulae*, Quart. Appl. Math. **4** (1949), 112–141.

[42] _____, *Cardinal spline interpolation*, CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 12, SIAM, 1973.

[43] L. L. Schumaker, *Spline funtions: Basic theory*, Pure and Applied Mathematics: A Wiley–Interscience Series of Texts, Monographs and Tracts, John Wiley & Sons, 1981.

[44] H. R. Schwarz, *Numerische Mathematik*, B. G. Teubner, Stuttgart, 1988.

[45] H. P. Seidel, *A new multiaffine approach to B–splines*, Comp. Aided Geom. Design **6** (1989), 23–32.

[46] L.E. Sigler, *Fibonacci's Liber Abaci. Leoanardo Pisano's book of calculation*, Springer, 2002.

[47] J. Stoer, *Einführung in die Numerische Mathematik I*, 4 ed., Heidelberger Taschenbücher, Springer Verlag, 1983.

[48] J. Stoer and R. Bulirsch, *Einführung in die Numerische Mathematik II*, 2 ed., Heidelberger Taschenbücher, Springer Verlag, 1978.

[49] K. Weierstraß, *Über die analytische Darstellbarkeit sogenannter willkührlicher Funktionen reller Argumente*, Sitzungsber. Kgl. Preuss. Akad. Wiss. Berlin (1885), 633–639,789–805.

[50] J. H. Wilkinson, *The evaluation of zeros of ill–conditioned polynomials I*, Numer. Math. **1** (1959), 150–166.

[51] ———, *The evaluation of zeros of ill–conditioned polynomials II*, Numer. Math. **1** (1959), 167–180.

[52] ———, *Error analysis of floating point computation*, Numer. Math. **2** (1960), 319–340.

[53] ———, *The perfidious polynomial*, Studies in Numerical Analysis (G. H. Golub, ed.), MAA Studies in Mathematics, vol. 24, The Mathematical Association of America, 1984, pp. 1–28.

Index 9