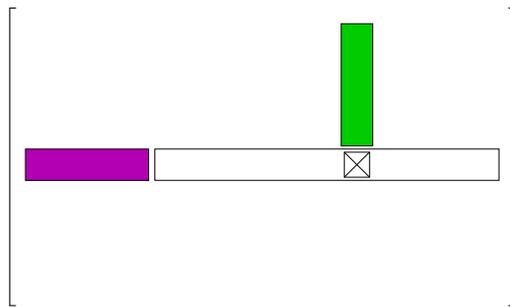
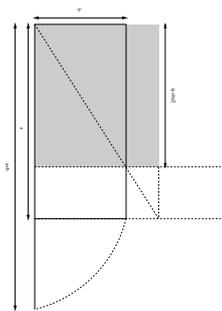


Einführung in die Numerische Mathematik

Vorlesung, zuerst gehalten im Wintersemester 2013/14

Tomas Sauer

Version 0.1
 Letzte Änderung: 24.8.2013



Es gibt sogenannte Mathematiker, die sich gerne ebenso für Gesandte der Weisheit gehalten 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

Da war alles so klar und einfach gesagt und doch nicht trocken; selbst in den mathematischen Entwicklungen verschwand nie der Endzweck des Ganzen; [...]

Kurd Laßwitz, *Aspira*

Reality is software. What does it matter what system its running on?

Rudy Rucker, *Postsingular*

It always does seem to me that I am doing more work than I should do. It is not that I object to the work, mind you; I like work: it fascinates me. I can sit and look at it for hours. I love to keep it by me: the idea of getting rid of it nearly breaks my heart.

Jerome K. Jerome, *Three men in a boat*

Don't ask me things I don't know. I can't tell you the answers. And don't ask me things I do know, because I won't tell you the answers.

Raymond Chandler, *The High Window*

Tomas Sauer
Lehrstuhl für Mathematik
mit Schwerpunkt Digitale Bildverarbeitung
Universität Passau
Innstr. 43
94032 Passau
Tomas.Sauer@uni-passau.de

Inhaltsverzeichnis | 0

1 Was will Numerische Mathematik?	4
1.1 Berechnung der Ableitung	5
1.2 Lösung eines linearen Gleichungssystems	5
1.3 Berechnung der Varianz	7
1.4 Aber sowas passiert doch nicht wirklich	11
2 Fehleranalyse	12
2.1 Gleitkommazahlen und -operationen	13
2.2 Fortpflanzung von Rundungsfehlern	21
2.3 Rückwärtsfehler	24
2.4 Summation	27
3 Lineare Gleichungssysteme und numerische Lösungen	30
3.1 Beispiel für lineare Gleichungssysteme	30
3.2 Normen und Konditionszahlen	31
3.3 Eine allgemeine Fehlerabschätzung	36
4 Direkte Methoden für lineare Gleichungssysteme	38
4.1 Dreiecksmatrizen	38
4.2 Naive Gauß–Elimination	40
4.3 Das fertige Lösungsverfahren	55
4.4 Fehleranalyse	56
4.5 Pivotsuche	58
4.6 Verbesserung der Genauigkeit – Skalierung	67
4.7 Verbesserung der Genauigkeit – Iteratives Verfeinern	68
5 Spezielle lineare Gleichungssysteme	72
5.1 Cholesky–Zerlegung	72
5.2 Bandierte Systeme	74
5.3 Total positive Systeme	76
6 Iterative Verfahren für lineare Gleichungssysteme	81
6.1 Der Spektralradius und ein allgemeines Konvergenzkriterium	83
6.2 Gauß–Seidel–Iteration und Jacobi–Iteration	87

7	Ausgleichsrechnung	94
7.1	Inverse Probleme – ein kurzer Exkurs	94
7.2	Nun aber überbestimmt	97
7.3	Least-squares-Lösungen und die Normalgleichungen	98
7.4	Singuläre-Werte-Zerlegung und die Pseudoinverse	101
7.5	Orthogonaltransformationen	105
7.6	Orthogonaltransformationen II – Givens-Rotationen	117
8	Polynome	121
8.1	Darstellungen polynomialer Kurven und Auswertungsverfahren	122
8.2	Interpolation – der Aitken-Neville-Algorithmus	129
8.3	Interpolation – der Newton-Ansatz	133
8.4	Approximationsgüte von Interpolationspolynomen	141
8.5	Die schlechte Nachricht	145
9	Splines	149
9.1	Der Algorithmus von de Boor	150
9.2	Splines und Interpolation	154
9.3	Eigenschaften der B-Splines	162
9.4	Der Splineraum	165
9.5	Knoteneinfügen	170
9.6	Die Schoenberg-Whitney-Bedingung	172
10	Nichtlineare Gleichungen	174
10.1	Terminologie	176
10.2	Das Bisektionsverfahren	176
10.3	Regula falsi und das Sekantenverfahren	179
10.4	Das Newton-Verfahren und Fixpunktiterationen	184
11	Numerische Integration und orthogonale Polynome	192
11.1	Quadraturformeln	193
11.2	Klassische Formeln	197
11.3	Zusammengesetzte Quadraturformeln	199
11.4	Gauß-Quadratur	203
11.5	Orthogonale Polynome	207
A	Anhang	210
A.1	Rückwärtsfehler der Gauß-Elimination	210
A.2	Nachiteration	213
A.3	Der Satz von Weierstrass – Beweis	216
A.4	Schoenberg-Whitney und die totale Nichtnegativität der Kollokationsmatrix	217
A.5	Eine Anwendung nichtlinearer Gleichungen: Das Bierkastenproblem	223
A.6	Nullstellen von Polynomen	229

A.7 Quadratur nach Art des Gaußes 237

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

W. Gautschi

Was will Numerische Mathematik?

1

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 oder¹ $\text{RootOf}(x^{13} - 12x^7 + 9x^3 - 1)$ 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 nun 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.

¹Derartige Lösungen bekommt man oft bei sogenannten Computeralgebrasystemen, die *symbolische Rechnungen* durchführen. Das hat den Vorteil, daß die Ergebnisse immer zu 100% richtig sind, aber eben auch den Nachteil, daß sie manchmal eher nutzlos sind.

```

%% 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

1.1 Berechnung der Ableitung

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

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

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, bei der Limesbestimmung auch 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

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

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

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

Die Determinanten können wir nach der **Leibniz-Regel** berechnen:

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

wobei

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

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 , $j = 1, \dots, n$, nach (1.1).

Zu dieser „Lösungsmethode“ kann man einiges sagen:

- Der Test $a = 0$ setzt voraus, daß $\det A$ in einer *exakten* Arithmetik² 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

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

²Mit der auf Computern vorherrschenden nicht-exakten Arithmetik und deren Konsequenzen werden wir uns im nächsten Abschnitt beschäftigen.

Wir werden später sehen, daß „vernünftige“ Verfahren 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 $X = (x_1, \dots, x_N) \in \mathbb{R}^N$ ein Vektor von Zahlen (Meßwerten ...). Der Erwartungswert $E(X)$ ist definiert als

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

und die (empirische) **Standardabweichung** als

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

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 zwischenspeichern, 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:

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

```

%% 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).

Damit ergibt sich die Berechnungsvorschrift

$$S_0 = Q_0 = 0 \quad (1.5)$$

$$S_j = S_{j-1} + x_j, \quad Q_j = Q_{j-1} + x_j^2, \quad j = 1, \dots, n \quad (1.6)$$

und

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

Übung 1.1 Beweisen Sie die Formel (1.4). ◇

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

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.

- Wir betrachten $X = (5000, 5001, 5002)$. Man sieht leicht, daß $E(X) = 5001$ und $S(X) = 1$. Das Programm liefert aber³ den ziemlich falschen Wert $S(X) = 0.707107$.

³Auf (m)einem Pentium II unter Linux, SuSE Version 6.0 – also schon ein klein wenig veraltet in Zwischenzeit.

```
/*
  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).

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 Zahl⁴ ...*

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

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

mittels der Berechnungsregeln $M_0 = Q_0 = 0$ und

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

$$M_j = M_{j-1} + \frac{x_j - M_{j-1}}{j}, \quad j = 1, \dots, n. \quad (1.10)$$

Das Ergebnis ist dann

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

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

⁴Was ein klein wenig im Gegensatz zu (1.3) steht, wo lediglich nichtnegative Zahlen aufsummiert werden.

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

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

Fehleranalyse | 2

Eine besonders schöne, elementare und lesenswerte Einführung in das Wesen und Unwesen der Rundungsfehler und deren Konsequenzen ist die Arbeit von Goldberg [12] – in ihr findet sich 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. Als **absoluter Fehler** wird der Wert

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

2. als **relativer Fehler** (für $c \neq 0$) die Größe

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

bezeichnet.

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, wie sich mit der folgenden Übung leicht verifizieren lässt.

Übung 2.1 Zeigen Sie: wird die B -adische Zahl $x = .x_1x_2 \cdots x_\infty \times B^e$, $x_j \in \{0, \dots, B-1\}$, $e \in \mathbb{Z}$, durch $\hat{x} = .\hat{x}_1\hat{x}_2 \cdots \hat{x}_\infty \times B^e$, $\hat{x}_j \in \{0, \dots, B-1\}$, mit einem relativen Fehler von ε angenähert, so stimmen praktisch⁵ immer die ersten $-\log_B \varepsilon$ Stellen von x und \hat{x} überein. \diamond

⁵Es gibt ein paar mehr oder weniger pathologische Ausnahmen – diese kann man sich im Rahmen eines Lösungsversuchs ja herleiten!

2.1 Gleitkommazahlen und -operationen

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

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

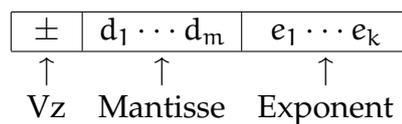
$$\pm \left(\sum_{j=1}^m d_j B^{-j} \right) B^e, \quad d_j \in \{0, \dots, B-1\}, e \in E.$$

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:



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 darstellbare⁶ 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**.

⁶So 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 2.4 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.

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

Der Name „Rundungsfehlereinheit“ kommt daher, daß dies gerade der relative Fehler ist, der auftritt, wenn man von einer darstellbaren reellen Zahl x zur nächstgelegenen Gleitkommazahl $\hat{x} \in \mathbb{F}$ übergeht. Wer ein Herz für Metriken hat, kann dies also als die (relative) **Distanz** zwischen \mathbb{F} und $D(\mathbb{F}) \subset \mathbb{R}$ bezeichnen.

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

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

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

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

Dieser Beweis ist einfach, irgendwie auch konstruktiv, aber keine praktische Rechenvorschrift, denn er erfordert die Kenntnis *aller* Stellen von x .

Für einen alternativen Beweis für *gerade*⁷ Werte von B setzen wir

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

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

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

(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

$$\varepsilon_{m+1} < \frac{B}{2} \quad \implies \quad \varepsilon_{m+1} \leq \frac{B}{2} - 1$$

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 müssen⁸. \square

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 $k > 1$? \diamond

⁷Man kann sich relativ leicht klarmachen, warum die Basis eine gerade Zahl sein muß: Nachdem man ja immer die B Ziffern $0, \dots, B-1$ 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 ...

⁸Und das funktioniert eben nur, wenn die Basis B eine gerade Zahl ist

Definition 2.7 Die Gleitkommaoperationen

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

werden wie folgt ausgeführt:

1. **Addition** (\oplus, \ominus): Schiebe die Zahl mit kleinerem Exponenten so lange nach rechts⁹, 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.

Der **Akkumulator** ist diejenige Gleitkommazahl, mit der der Rechner intern wirklich rechnet. Relevant ist natürlich nur die Mantissenlänge des Akkumulators.

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

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). Allerdings hat sich das mit der Division im Zeitalter von Arithmetik-Koprozessoren etwas überholt, inzwischen sind die Operationen alle in etwa gleich schnell und andere Dinge sind viel relevanter, beispielsweise das Speichermanagement.
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:

$$\begin{array}{r} .1230 \\ + .0306 \\ \hline .1536 \end{array} \rightarrow .154$$

⁹Diese Operation erhöht den Exponenten.

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

$$\begin{array}{r} .123 \\ + .030 \\ \hline .153 \end{array} \rightarrow .153$$

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

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

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

Das mit den „Phantasieziffern“ schreit danach, genauer erklärt zu werden. Natürlich ist $.123$ eine exakte Zahl, aber wenn vorher eine Rundung passiert ist¹⁰, dann kann dies das Rundungsergebnis einer beliebigen Zahl aus dem Intervall $[.1225, 1235)$ sein; entsprechend repräsentiert $.122$ dann auch das Intervall $[.1215, .1225)$ und die Differenz ist eine Zahl aus¹¹

$$[.1225, 1235) - [.1215, .1225) = (0, .002).$$

Der berechnete Wert $.001$ ist dann auch das beste, was man schaffen kann, nämlich der Mittelpunkt des Intervalls, im Sinne eines wirklich vor Rundung korrekten Ergebnisses kann aber ein relativer Fehler von bis zu 100% auftreten! Und das ist nicht schön.

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

Beispiel 2.10 (IEEE 754) Die in heutigen¹² PC-Prozessoren verwendete Gleitkommaarithmetik entspricht dem IEEE¹³ 754-Standard. Die dabei verwendete Basis ist 2 und die arithmetischen Datentypen sind als

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 NaNs¹⁴.

Der Standard IEEE 854 lässt übrigens als Basis 2 und 10 zu!

¹⁰Was man bei der Rechnung selbst ja nie weiss.

¹¹Für die Subtraktion von Mengen, in unserem Falle von Intervallen, verwenden wir hier die durchaus gebräuchliche Notation $A - B = \{a - b : a \in A, b \in B\}$.

¹²Das ist inzwischen überholt, aber wozu immer wieder anpassen? Inzwischen ist IEEE 854 aktuell. Mindestens.

¹³IEEE = Institute of Electrical and Electronical Engineers

¹⁴Not a Number, Ergebnisse von unzulässigen Operationen wie Wurzeln aus negativen Zahlen

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

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

also ist

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

Das berechnete Ergebnis hingegen ist

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

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,

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

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:

$$\begin{array}{r|cccc|c|l} & .x_1 & x_2 & \cdots & x_m & 0 & 0 \dots 0 \\ k=0 & .y_1 & y_2 & \cdots & y_m & 0 & 0 \dots 0 \\ k=1 & .0 & y_1 & \cdots & y_{m-1} & y_m & 0 \dots 0 \end{array}$$

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

$$x \ominus y = rd_m(x - \hat{y}) = x - \hat{y} - \delta,$$

wobei¹⁵ $|\delta| \leq \frac{B}{2} B^{-m-1} \leq \frac{1}{2} B^{-m}$. 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

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

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

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

¹⁵Achtung: 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.

also

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

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 $x - \hat{y} = B^{-1} = .10 \dots 0$ 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

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

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: ist¹⁶ $x \cdot y \neq 0$ 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.

¹⁶Eine 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

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

was zu der pessimistischen Abschätzung

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

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

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

was die wesentliche bessere Fehlerschranke

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

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

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

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. Dieses berechnen wir nach der Regel¹⁷

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

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 $j = 2, \dots, n$ gilt

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

wobei $\varepsilon_1 = 0$.

¹⁷Wir sind in der Numerik und hier spielt es eine wesentliche Rolle, wie man die Dinge berechnet. Andere Formeln produzieren dann auch andere Rundungsfehler.

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

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

□

Damit erhalten wir den absoluten Fehler

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

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

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

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

$$\lim_{u \rightarrow 0} \frac{\psi(u)}{\phi(u)} = 0. \quad (2.9)$$

Da der Rundungsfehler \hat{u} immer als sehr klein (im Vergleich zu n) angenommen wird, also $n\hat{u} < 1$, werden wir $O((n\hat{u})^2)$ -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 $x^T y$, $x, y \in \mathbb{R}^n$, erfüllt die Ungleichung

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

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 $|x_1 y_1| + \dots + |x_n y_n|$, 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

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

Wir interpretieren Fehler aus der Berechnung also jetzt als Fehler in den Eingabedaten. Diese Idee, die auf Wilkinson [46] 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 $f : D \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$ eine beliebige Funktion und $\hat{f} : D \rightarrow \mathbb{R}^m$ ein numerisches Verfahren dazu. Ein **Rückwärtsfehler** ρ des Verfahrens ist eine 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

$$\frac{\|x - \hat{x}\|}{\|x\|} \leq \rho \quad (2.11)$$

erfüllt ist. Ein **komponentenweiser Rückwärtsfehler** $\rho \in \mathbb{R}^n$ ist analog als kleinste Fehler-schranke in

$$\frac{|x_j - \hat{x}_j|}{|x_j|} \leq \rho_j, \quad j = 1, \dots, n, \quad (2.12)$$

definiert.

Es versteht sich von selbst, daß man versucht, möglichst kleine Werte für ρ zu bestimmen, da sonst die Aussagekraft nur sehr limitiert ist.

Schematisch ist die Idee des Rückwärtsfehlers wie folgt:

$$\begin{array}{ccc} x & \rightarrow & \hat{f} \\ \updownarrow & & \searrow \\ \hat{x} & \rightarrow & f \end{array} \quad \hat{y} = \hat{f}(x)$$

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

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

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

bzw.

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

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

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

Ist die Bilinearform D^2f 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.

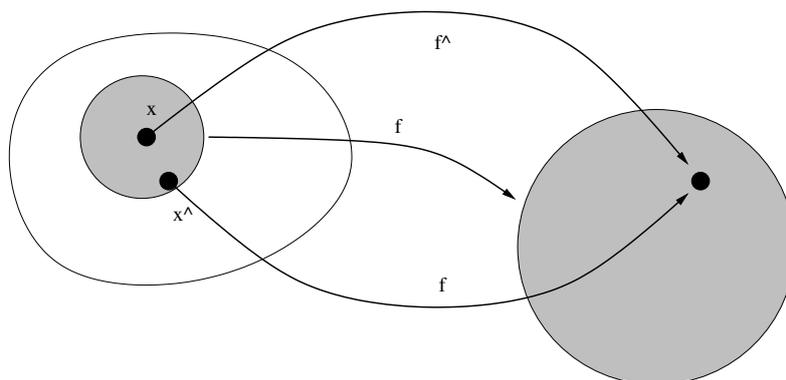


Abbildung 2.1: Rückwärtsfehler und Konditionszahl. Zuerst findet man ein (natürlich möglichst gutes) \hat{x} so daß $\hat{f}(x) = f(\hat{x})$, der Rückwärtsfehler ist dann der Radius eines (möglichst kleinen) Kreises um x , der auch \hat{x} enthält. Dieser Kreis wird nun durch f auf ein anderes Gebiet (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

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

Also ist

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

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 *Überschätzung* des Fehlers neigen. Bessere, das heißt schärfere Resultate erhält man zumeist durch die sogenannte **run-nig error analysis**¹⁸, oder über statistische Methoden, die dann von einer gewissen Verteilung der Eingangsdaten ausgehen und normalerweise Aussagen über mittlere Fehler liefern.

2.4 Summation

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

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

zur Berechnung von $x^T \mathbf{1}_n = x_1 + \dots + x_n$, $\mathbf{1}_n = (1, \dots, 1)^T$ als Spezialfall des Skalarprodukts die Fehlerabschätzung

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

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 Fehler-schranke verbessern?“ Die Antwort ist die **compensated summation** von Kahan. Der Algorithmus ist allerdings etwas aufwendiger:

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

Die Idee dieses Verfahrens ist, daß die Folge c_j die Rundungsfehler aufsammelt und immer wieder in die Rechnung steckt und auf diese Art letztendlich kompensiert, so daß man ein wesentlich besseres Fehlerverhalten bekommt. Den Satz beweisen wir nicht, wer sich für den voll und ganz nichttrivialen Beweis interessiert, findet diesen in [12].

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

$$s_n = \sum_{j=1}^n x_j (1 + \delta_j) + O(n \hat{u}^2) \sum_{j=1}^n |x_j|, \quad |\delta_j| \leq 2 \hat{u}, \quad j = 1, \dots, n. \quad (2.15)$$

¹⁸Zumindest der Name sollte einmal gefallen sein.

```
%% 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

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

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

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

$\frac{1}{4}$ Breite und Länge zusammen sind 7
Handbreiten, Länge und Breite
zusammen sind 10 Handbreiten.

Babylonisches Gleichungssystem in
zwei Unbekannten, Susa,
2. Jahrtausend v. Chr

Lineare Gleichungssysteme und numerische Lösungen

3

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

$$Ax = b \quad (3.1)$$

erfüllt ist. 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. Gleichungssysteme, die aus mehr Gleichungen als Variablen bestehen, also $m > n$ erfüllen, nennt man **überbestimmt**, Gleichungssysteme mit „zu wenigen“ Gleichungen, $m < n$, hingegen **unterbestimmt**. 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 Beispiel für lineare Gleichungssysteme

Das einfachste Auftreten von linearen Gleichungssystemen in der „Praxis“ ist im Kontext der **Interpolation**: Es sei \mathcal{F}_n ein n -dimensionaler Vektorraum und f_1, \dots, f_n eine Basis von \mathcal{F}_n . Außerdem seien Punkte x_1, \dots, x_n gegeben. Das **Interpolationsproblem** besteht nun darin, zu vorgegebenen Werten y_1, \dots, y_n eine Funktion $f = \sum_{j=1}^n a_j f_j$ zu finden, so daß

$$f(x_j) = y_j, \quad j = 1, \dots, n.$$

Anders gesagt,

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

oder eben

$$[f_j(x_k) : j, k = 1, \dots, n]^T a = y. \quad (3.2)$$

Mit ein klein wenig Phantasie kann man das auch auf beliebige lineare Funktionale und Operatoren auf endlichdimensionalen Vektorräumen erweitern. Die Idee ist einfach, verwirrt aber trotzdem gerne.

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 [13, 17], aus denen auch die Kapitel über numerische Lineare Algebra entnommen sind.

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 $\|\cdot\| : \mathbb{R}^n \rightarrow \mathbb{R}$ heißt (Vektor-)Norm, wenn gilt

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 oder Manhattan-Norm

$$\|x\|_1 = \sum_{j=1}^n |x_j|.$$

2. Euklidischer Abstand

$$\|x\|_2 = \sqrt{\sum_{j=1}^n |x_j|^2}.$$

3. Die ∞ -Norm, die den Worst-Case-Fehler

$$\|x\|_{\infty} = \max_{j=1, \dots, n} |x_j|$$

angibt.

4. Das sind alles Spezialfälle der sogenannten p -Normen

$$\|x\|_p = \left(\sum_{j=1}^n |x_j|^p \right)^{1/p}, \quad 1 \leq p \leq \infty. \quad (3.3)$$

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

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

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

Analog zu Vektornormen kann man auch Matrixnormen definieren, indem man 3.1 einfach wörtlich übernimmt. Möglich ist dies, weil die Matrizen ebenfalls einen **Vektorraum** bilden, also addiert und mit Zahlen aus \mathbb{R} multipliziert werden können. Und Normen gehören immer zu Vektorräumen.

Definition 3.3 Eine Abbildung $\|\cdot\| : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$ heißt **Matrix-Norm**, wenn

1. $\|A\| \geq 0$ und $\|A\| = 0$ dann und nur dann, wenn $A = 0$.
2. $\|cA\| = |c| \|A\|$, $c \in \mathbb{R}$.
3. $\|A + B\| \leq \|A\| + \|B\|$.

Beispiel 3.4 (Matrixnormen) Jetzt 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** und ist ausgesprochen beliebt.

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

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

Diese Norm misst, wie stark die Matrix Vektoren „vergrößern“ kann.

3. Beispiele für die Operatornormen zu den p -Normen:

Spaltensummennorm:

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

Zeilensummennorm:

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

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

Die nächste Definition geht nun über den Vektorraum hinaus und berücksichtigt außerdem die Tatsache, daß Matrizen, im Gegensatz zu „normalen“ Vektoren¹⁹ auch multipliziert werden können, also eine (nichtkommutative) Algebra bilden. Und vernünftiges Verhalten der Norm bezüglich dieser Multiplikation gibt es nicht immer und muss daher gefordert werden.

Definition 3.5 (Konsistenz & 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:

1. Jede Operatornorm ist konsistent und mit der zugehörigen Vektornorm verträglich.

2. Die Norm

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

ist nicht konsistent.

¹⁹Weil wir gerade dabei sind: Was ist eigentlich ein Vektor?

3. 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 nach Definition 2.22 erwarten.

Satz 3.7 Für jede Operatornorm $\|\cdot\|$ gilt

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

Die Konditionszahl $\kappa(A)$ misst also die Konditioniertheit der Abbildung $f(A) = A^{-1}$.

Für den Beweis brauchen wir erst eine Hilfsaussage. Da zu jeder Operatornorm per Definitionem auch eine Vektornorm gehört, bezeichnen wir jetzt beide Normen einfach mit $\|\cdot\|$ ²⁰

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: Zu x existiert immer ein **dualer Vektor** x' , der durch die Eigenschaften

$$x^T x' = 1 \quad \text{und} \quad \max_{\|z\|=1} |z^T x'| = 1$$

definiert ist, und zwar nach **Satz von Hahn–Banach**²¹. Dann setzen wir lediglich $A = yx'^T$ und es ist $Ax = y$ ($x'^T x$) = y und

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

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

²⁰Es ist ein schöner Verständnistest, im Rest des Beweises die Unterscheidungen $\|\cdot\|_V$ und $\|\cdot\|_M$ für Vektor- und Operatornorm dazuzuschreiben. Viel Spass dabei!

²¹Das ist ein durchaus nichttriviales Resultat aus der Funktionalanalysis, das hier nur zitiert, nicht bewiesen werden soll. Interessenten sei das Buch [23] ans Herz gelegt, das eine sehr schöne Einführung in die Funktionalanalysis bietet.

Beweis von Satz 3.7: Aus der (formalen²²!) Entwicklung

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

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

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

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

$$\sup_{\|E\| \leq 1} \|A^{-1}EA^{-1}\| = \|A^{-1}\|^2, \quad (3.5)$$

denn dann ist

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

woraus (3.4) folgt. Nun ist „ \leq “ in (3.5) einfach:

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

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

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

(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

$$\|A^{-1}EA^{-1}\| = \max_{\|z\|=1} \|A^{-1}EA^{-1}z\| \geq \|A^{-1}EA^{-1}y\| = \|A^{-1}Ex\| \|A^{-1}\| \quad (3.6)$$

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

$$\|A^{-1}Ex\| = \|A^{-1}y\| = \|A^{-1}\|. \quad (3.7)$$

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

²²Das 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

$$\widehat{A} \widehat{x} = b \quad (3.8)$$

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*

$$\|A^{-1}E\| = \rho < 1 \quad (3.9)$$

sowie

$$\|E\| \leq \delta \|A\|. \quad (3.10)$$

Dann gilt für den relativen Fehler die Abschätzung

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

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

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

sein soll. In der Tat, wenn man den „Extremfall“ $\delta := \|E\| / \|A\|$ annimmt²³, dann ist auch

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

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 überprüft²⁴ 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} .

²³Was für $A \neq 0$ ja kein Problem ist.

²⁴Allerdings wird dort **nicht** die „exakte“ Konditionszahl bestimmt, was eine $O(n^3)$ -Geschichte wäre, sondern „nur“ eine Näherung, die mit einem Aufwand von $O(n^2)$ 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 $\hat{A} = A - E$ invertierbar und es gilt*

$$\|\hat{A}^{-1}\| \leq \frac{\|A^{-1}\|}{1 - \rho}. \quad (3.13)$$

Beweis: Wir betrachten

$$\hat{A} = A - E = A (I - A^{-1}E)$$

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

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

Da

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

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

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

weil die Reihe auf der rechten Seite absolut konvergiert:

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

Also ist

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

□

Beweis von Satz 3.9: Es ist

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

also, mittels (3.10) und (3.13)

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

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 ist, läßt sich bei genauerer Betrachtung 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.*

Vergebens predigt Salomo
Die Leute machens doch nicht so.

Wilhelm Busch

Direkte Methoden für lineare Gleichungssysteme

4

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 $A = [a_{jk} : j, k = 1, \dots, n] \in \mathbb{R}^{n \times n}$ heißt *untere Dreiecksmatrix* 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. \diamond

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

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

eine untere Dreiecksmatrix, dann lautet das zugehörige Gleichungssystem in expliziter Form

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

was sofort

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

```

%% VorElim.m (Numerik 1)
%% -----
%% Vorwaartselimination, 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

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

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

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

heißt **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*²⁵. 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 gilt natürlich auch für die Rücksubstitution.

²⁵Für *floating point operations*

```

%% 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

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

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. Wir werden sehen, daß das Zerlegen der Matrix der deutlich aufwendigere Teil des Lösungsverfahrens ist.

Bevor wir das entsprechende 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 , $m = 1, \dots, n$, die m -te **Hauptuntermatrix**

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

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. $\det A_m \neq 0$, $m = 1, \dots, n$. 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

$$A = LU \quad \text{und} \quad \ell_{jj} = 1, \quad j = 1, \dots, n. \quad (4.3)$$

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

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

Generell sollte man aber von Determinanten **immer** die Finger lassen, da sie so ziemlich zum instabilsten gehören, was man in der numerischen linearen Algebra berechnen kann.

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

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

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

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

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

□

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 äquivalent zu

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

und da $\ell_{jr} = 0$ falls $r > j$ und $u_{rk} = 0$ falls $r > k$, heißt das, daß von der Summe nicht mehr so viel übrigbleibt:

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

Aus dieser Identität 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

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

und

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

kennen und daß die beiden sich passend multiplizieren:

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

Das * unten rechts bedeutet, daß die Matrix hier (noch) machen darf, was sie will. Ausserdem beachten wir, daß diese Forderung für $m = 1$ *trivialerweise* erfüllt ist, weil es ja gar keine Bedingung an die Matrix gibt. 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

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

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: es muß $u_{mm} \neq 0$ sein.

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

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

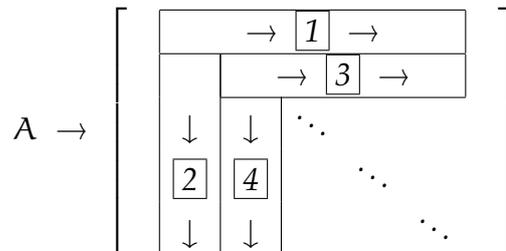
liefert. Außerdem ist das Ganze so gebaut, daß jetzt (4.6) auch mit m anstelle von $m - 1$ gilt. □

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

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

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



```
%% 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
end
% endfunction
```

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

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

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

bestimmen. Also erhalten wir die erste Zeile unserer „Ergebnismatrix“ LU^{26} als

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

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

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

und somit

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

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

$$u_{2k} = a_{2k} - \sum_{r=1}^1 \ell_{2r} u_{rk}, \quad \Rightarrow \quad \begin{aligned} u_{22} &= a_{22} - \ell_{21} u_{12} = -\frac{1}{2}, \\ u_{23} &= a_{23} - \ell_{21} u_{13} = a_{23} = 1. \end{aligned}$$

Tragen wir das ein, so erhalten wir

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

²⁶Im „oberen“ Dreieck dieser Matrix, die Diagonale eingeschlossen, findet sich U, im „unteren“ Dreieck die Matrix L. Da deren Diagonalelemente ℓ_{jj} ja sowieso alle den Wert 1 haben, brauchen wir für sie keinen Speicherplatz zu vergeuden.

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

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

erhält, ergo

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

Wie sich die Zerlegung schließlich zu

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

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

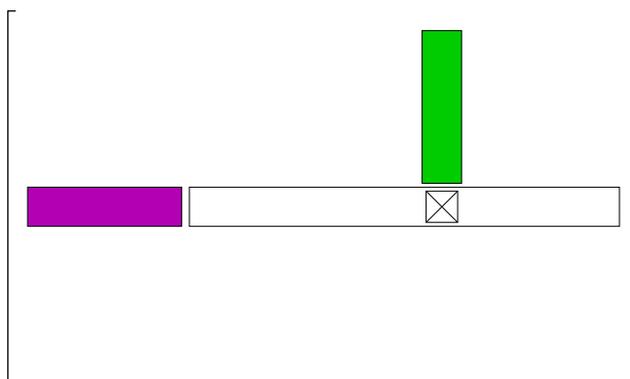


Abbildung 4.1: Schematische Darstellung des Doolittle-Verfahrens. Bei der Bestimmung der m -ten Zeile wird der neue Eintrag dadurch bestimmt, daß man vom „Originalwert“ der Matrix A das innere Produkt der beiden (farbigen) $(m-1)$ -Vektoren am Anfang der Zeile und der Spalte, in denen sich der Eintrag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

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

```
%% 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.

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 $l_{jj} = 1$), 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 $m = 1, \dots, n$, 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 $O(n^3)$, das wird bei der Gauß-Elimination auch so sein. Dies ist eine Größenordnung mehr als bei der Vorwärtselimination und der Rücksubstitution.

Nun zur Gauß-Elimination.

Definition 4.11 Eine Gauß-Transformation oder Gauß-Matrix ist eine Matrix der Form

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

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 $I + ye_k^T$:

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

3. Für eine Matrix A in Zeilenform

$$A = \begin{bmatrix} \mathbf{a}_1^\top \\ \vdots \\ \mathbf{a}_n^\top \end{bmatrix}$$

mit Zeilenvektoren $\mathbf{a}_1^\top, \dots, \mathbf{a}_n^\top$ ist

$$M_k A = \begin{bmatrix} \mathbf{a}_1^\top \\ \vdots \\ \mathbf{a}_k^\top \\ \mathbf{a}_{k+1}^\top - y_{k+1} \mathbf{a}_k^\top \\ \vdots \\ \mathbf{a}_n^\top - y_n \mathbf{a}_k^\top \end{bmatrix} = A - y \mathbf{a}_k^\top,$$

das heißt, von den Zeilen $\mathbf{a}_j^\top, j = k+1, \dots, n$, wird das y_j -fache der Zeile \mathbf{a}_k^\top abgezogen.

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

$$M_j = I - y^{(j)} \mathbf{e}_j^\top, \quad j = 1, \dots, n-1,$$

zu finden, so daß

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

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

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

Dazu konstruieren wir induktiv die Gauß-Transformationen M_1, M_2, \dots so, daß

$$M_m \cdots M_1 A = \begin{bmatrix} U_m & B_m \\ 0 & \tilde{A}_m \end{bmatrix}, \quad m = 0, \dots, n-1, \quad (4.10)$$

wobei

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

und $(U_m)_{jj} \neq 0, j = 1, \dots, m$. 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

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

```
%% 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.

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

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 zufällige²⁷, 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:

```
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 Matrix²⁸ $x * y$ von A abziehen:

²⁷Das ist natürlich bei jedem Ausprobieren eine andere Matrix, also nicht erschrecken!

²⁸Spalte mal Zeile gibt eine Matrix!

```
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
    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 =
```

```

%% 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.

```

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.

```

%% 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).

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**

$$A = LU \quad (4.13)$$

mittels Gauß–Elimination oder nach Doolittle.

2. Bestimme die Lösung y von

$$Ly = b \quad (4.14)$$

durch **Vorwärtselimination**.

3. Bestimme die Lösung x von

$$Ux = y \quad (4.15)$$

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 Vorwärtselimination so modifiziert, daß sie alle Diagonalelemente als 1 annimmt, wie in Programm VorElim1.m gezeigt.

```

%% VorElim1.m (Numerik 1)
%% -----
%% Vorwaartselimination, 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 Diagonalelemente von L den Wert 1 haben.

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

$$|A| = [|a_{jk}| : j, k = 1, \dots, n].$$

Für Matrizen definieren wir die **Halbordnung**²⁹ „ \leq “ als

$$A \leq B \quad \Leftrightarrow \quad a_{jk} \leq b_{jk}, \quad j, k = 1, \dots, n.$$

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

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

Beweis: Für $1 \leq j, k \leq n$,

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

□

²⁹Zur 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 *Halbordnung* muß dies nicht der Fall sein.

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 $O(\hat{u}^2)$ anstelle des (korrekteren) $O(n\hat{u}^2)$

Wenden wir nun das Verfahren (4.13)–(4.15) an, dann erhalten wir also zuerst eine näherungsweise LU-Zerlegung \hat{L} und \hat{U} , dann eine Näherungslösung \hat{y} (die aus \hat{L} , nicht aus L berechnet wird!) und schließlich ein berechnetes \hat{x} (das aus \hat{U} und \hat{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 \hat{A} , so daß

$$\hat{A} \hat{x} = b \quad (4.16)$$

und

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

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 erste³⁰ 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 $\hat{x} = \hat{f}(A, b)$ liefert. Nun schieben wir alle Rundungsfehler auf den ersten Parameter A und bestimmen eine Matrix \hat{A} , so daß $\hat{x} = f(\hat{A}, b)$ ist, und dann schätzen wir ab, inwieweit sich A und \hat{A} unterscheiden – das ist gerade (4.17). Daß alle Fehler ausschließlich auf A geschoben werden ist zuerst einmal total *willkürlich*³¹, 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“.

³⁰Und glücklicherweise auch einzige.

³¹Aber das ist die Auswahl von Sündenböcken eigentlich immer

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.

Der Beweis selbst findet sich in Anhang A.1.

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 4.23 Für eine Vektornorm $\|\cdot\|$ und eine konsistente, mit der Vektornorm verträgliche und „vernünftige“³² Matrixnorm sei γ so gewählt, daß

$$\left\| \begin{array}{c} \hat{L} \\ \hat{U} \end{array} \right\| \leq \gamma \|A\|. \quad (4.18)$$

Dann ist

$$\frac{\|\hat{x} - x\|}{\|x\|} \leq \frac{n\hat{u}(3 + 5\gamma)}{1 - n\hat{u}(3 + 5\gamma)} \kappa(A), \quad x \neq 0, \quad (4.19)$$

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

Bemerkung 4.24 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

$$\begin{vmatrix} \hat{L} \\ \hat{U} \end{vmatrix} \geq |\hat{L}\hat{U}| = |\hat{A}|$$

wird γ in (4.18) einen Wert > 1 haben.

Leider kann der Wert γ sehr schnell sehr groß werden.

Beispiel 4.25 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

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

³²Der korrekte Begriff heisst **monotone Matrixnorm**: $|A| \leq |B| \Rightarrow \|A\| \leq \|B\|$

Dann ist

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

aber

$$|\hat{L}| \quad |\hat{u}| = \begin{bmatrix} .001 & 1 \\ 1 & 2000 \end{bmatrix}$$

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.

Definition 4.26 Das Element a_{11} einer Matrix $A \in \mathbb{R}^{n \times n}$ bezeichnet man als **Pivotelement**.

Definition 4.27 Eine Matrix $P \in \mathbb{R}^{n \times n}$ heißt **Permutationsmatrix**, wenn

$$P \in \{0, 1\}^{n \times n} \quad \text{und} \quad \sum_{r=1}^n p_{rk} = \sum_{r=1}^n p_{jr} = 1, \quad j, k = 1, \dots, n. \quad (4.20)$$

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

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

Wir bezeichnen mit

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

die Menge aller Vertauschungsmatrizen. Diese erzeugen dann auch die (multiplikative) Gruppe³³ der Permutationsmatrizen.

³³Jede Permutation hat eine Inverse und das Produkt zweier Permutationen ist wieder eine Permutation.

Übung 4.2 Zeigen Sie:

1. Jede Permutationsmatrix P ist als Produkt von Vertauschungsmatrizen darstellbar und jedes Produkt

$$\prod_{j=1}^k P_j, \quad P_j \in \Pi_n, k \in \mathbb{N},$$

ist wieder eine Permutationsmatrix.

2. Die Menge der Permutationsmatrizen bildet eine Gruppe.

◇

Bemerkung 4.28 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 $P \in \Pi_n$ – eine lineare Abbildung mit dieser Eigenschaft bezeichnet man als **Projektion**.

Damit haben wir das allgemeine Resultat zur Gauß–Elimination.

Satz 4.29 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

$$PA = LU. \quad (4.21)$$

Korollar 4.30 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 $(PA)_m$, $m = 1, \dots, n$, von PA invertierbar sind.

Beweis: Korollar 4.7 und Satz 4.29. □

Bemerkung 4.31 (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 können³⁴, ist die Fehleranalyse von (4.21) exakt dieselbe wie die für die naive Gauß–Elimination.

³⁴Es wird ja nichts gerechnet, nur umgestellt, und wenn dabei Rundungsfehler auftreten würden, dann wäre etwas ganz gewaltig faul.

3. Satz 4.29 ergibt als Verfahren die **Gauß–Elimination mit Spaltenpivotsuche**, alternativ könnte man auch Zerlegungen

$$\begin{array}{ll} \text{Zeilenpivotsuche} & \text{AQ} = \text{LU}, \quad \text{Q} \in \Pi_n, \\ \text{Totalpivotsuche} & \text{PAQ} = \text{LU}, \quad \text{P, Q} \in \Pi_n \end{array} \quad (4.22)$$

untersuchen. Die zugehörigen Lösungsmethoden sind

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

4. Aus $\det P = 1$ und (4.21) folgt, daß

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

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

Beweis von Satz 4.29: 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 $P_j \in \Pi_n, j = 1, \dots, n$, so daß

$$A^{(m)} := M_m P_m \cdots M_1 P_1 A = \begin{bmatrix} U_m & B_m \\ 0 & \tilde{A}_m \end{bmatrix}, \quad m = 0, \dots, n, \quad (4.23)$$

wobei U_m, B_m und \tilde{A}_m wie in (4.11) sind.³⁵ 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 \tilde{A}_m und wählen k so, daß

$$|\tilde{a}_{k1}| \geq |\tilde{a}_{j1}|, \quad j = 1, \dots, n - m. \quad (4.24)$$

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

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

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

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

³⁵Dies ist genau das Gegenstück zu (4.10).

sowie

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

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

$$\left| y_j^{(m+1)} \right| \leq 1. \quad (4.25)$$

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

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

so daß die Zerlegung aus (4.23) auf den Fall $m+1$ erweitert ist. Nach n Schritten haben wir dann also, daß

$$M_n P_n \cdots M_1 P_1 A = U, \quad M_n = P_n = I, \quad (4.26)$$

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

$$(M_{n-1} P_{n-1} \cdots M_1 P_1)^{-1} = P_1 M_1^{-1} \cdots P_{n-1} M_{n-1}^{-1} = PL, \quad (4.27)$$

denn dann ist

$$P^{-1} A = P^{-1} P_1 M_1^{-1} \cdots P_{n-1} M_{n-1}^{-1} U = P^{-1} P L U = L U \quad (4.28)$$

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

Bleibt also nur noch (4.27); das funktioniert, weil wir Gauß-Transformationen und Permutationen in einem gewissen Sinne vertauschen können, solange nur die Indizes zusammenpassen. Dazu schreiben wir $M_j = M_j(y^{(j)})$ und verwenden die Identität

$$M_j(y^{(j)}) P[k, \ell] = P[k, \ell] M_j(P[k, \ell] y^{(j)}), \quad 1 \leq j < k < \ell \leq n, \quad (4.29)$$

aus der (4.27) unmittelbar folgt. Aber auch (4.29) ist nicht schwer zu beweisen: unter Verwendung der Kurzschreibweise $y = y^{(j)}$, $M = M_j(y^{(j)})$ und $P = P[k, \ell]$ erhalten wir, daß

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

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

Beispiel 4.32 Wir betrachten³⁶ 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

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

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

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

wobei

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

³⁶Es wurde mal irhendwann in der Vorlesung ein Beispiel „mit Zahlen“ gewünscht.

Die entscheidende Aktion der „neuen“ Gauß–Elimination ist die **Spaltenpivotsuche** in (4.24): man bestimmt eine Permutation $P_{m+1} = P[m+1, k]$, $k \geq m+1$, so daß

$$\left| \left(P_{m+1} A^{(m)} \right)_{m+1, m+1} \right| \geq \left| \left(P_{m+1} A^{(m)} \right)_{j, m+1} \right|, \quad j = m+2, \dots, n. \quad (4.30)$$

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

$$\left| \left(A^{(m)} Q_{m+1} \right)_{m+1, m+1} \right| \geq \left| \left(A^{(m)} Q_{m+1} \right)_{m+1, j} \right|, \quad j = m+2, \dots, n, \quad (4.31)$$

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

$$\left| \left(P_{m+1} A^{(m)} Q_{m+1} \right)_{m+1, m+1} \right| \geq \left| \left(P_{m+1} A^{(m)} Q_{m+1} \right)_{j, j'} \right|, \quad j, j' = m+2, \dots, n. \quad (4.32)$$

Wichtig ist, daß der Suchaufwand bei der Totalpivotsuche wesentlich höher ist, nämlich quadratisch in jedem Schritt, und deswegen mit einem $O(n^3)$ zur Komplexität des Verfahrens beiträgt. Da das dieselbe Größenordnung wie die Zerlegung selbst ist und da Vergleiche nicht wesentlich billiger sind als arithmetische Operationen, vom Speicherzugriff ganz zu schweigen³⁷.

Die Gauß–Elimination *mit* Spaltenpivotsuche 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 4.33 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 \hat{A} , so daß

$$\hat{A} \hat{x} = b \quad (4.33)$$

und

$$\left| \hat{A} - A \right| \leq n \hat{u} \left(3|A| + 5 \left| \hat{L} \right| \left| \hat{U} \right| \right) + O(\hat{u}^2). \quad (4.34)$$

Bemerkung 4.34 (Beliebte Mißverständnisse)

1. „Pivotsuche liefert immer genauere Ergebnisse“

Die Matrix

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

³⁷Heutzutage (2013) dauern tatsächlich Speicherzugriffe oftmals länger als arithmetische Berechnungen, den Arithmetikkomponenten der Prozessoren sei dank.

```

%% 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 mittels Gauß-Elimination mit Spaltenpivotsuche.

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

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

liefert. Durch 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

$$\begin{bmatrix} \frac{2}{10} & 0 & 0 \\ 1 & 100 & 100 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ \frac{10}{9} & 1 & \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{10} & 0 & 0 \\ 100 & 100 & \\ & & 1 \end{bmatrix}.$$

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

$$\begin{bmatrix} 1 & 100 & 100 \\ \frac{2}{10} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ \frac{2}{10} & 1 & \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 100 & 100 \\ -90 & -90 & \\ & & 1 \end{bmatrix}$$

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 $D_1, D_2 \in \mathbb{R}_+^{n \times n}$

$$Ax = b \quad \Leftrightarrow \quad \underbrace{D_1 A D_2^{-1}}_{=:B} \underbrace{D_2 x}_{=:y} = \underbrace{D_1 b}_{=:c} \quad \Leftrightarrow \quad By = c. \quad (4.35)$$

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

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

mit Inverser

$$A^{-1} = \begin{bmatrix} -\frac{1}{999} \alpha & \frac{1000}{999} \\ \frac{1}{999} \alpha & -\frac{1}{999} \end{bmatrix} \quad \Rightarrow \quad \|A^{-1}\|_\infty = \frac{1000 + \alpha^{-1}}{999}.$$

Damit ist

$$\kappa_\infty(A) = \begin{cases} \frac{1001}{999} (1000 \alpha + 1), & \alpha \geq 2/1001, \\ \frac{2000 + 2\alpha^{-1}}{999}, & \alpha < 2/1001, \end{cases} \quad (4.36)$$

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 optimale³⁸ 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

$$\frac{|a_{jm}^{(m)}|}{\max_{r=m,\dots,n} |a_{jr}^{(m)}|} = \max_{k=m,\dots,n} \frac{|a_{km}^{(m)}|}{\max_{r=m,\dots,n} |a_{kr}^{(m)}|}. \quad (4.37)$$

Satz 4.36 (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 (4.37) 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 $|\hat{r}_1| \leq n\hat{u}|1_n|$ ³⁹, 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

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

was wir solange fortsetzen bis (hoffentlich) irgendein r_k klein genug wird.

Iteratives Verfeinern⁴⁰ alias **iterative refinement** berechnet also, ausgehend von $\hat{x}_0 = \hat{x}$, eine Folge

$$\hat{r}_j = b \ominus A \otimes \hat{x}_{j-1}, \quad \hat{x}_j = \hat{x}_{j-1} \oplus \text{Solve}(Ax = \hat{r}_j), \quad j \in \mathbb{N}, \quad (4.38)$$

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

³⁸Aber gar nicht so schlechte ...

³⁹Die Lösung \hat{x} ist also bis auf unvermeidbare Rundungsfehler exakt!

⁴⁰Auch als **Nachiteration** bekannt.

```

%% 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 textttk–facher Nachiteration.

Bemerkung 4.37 Beim „klassischen“ iterative refinement werden die Operationen aus (4.38) in doppelter Genauigkeit, also mit einem relativen Fehler von höchstens $2\hat{\epsilon}^2$, berechnet.

Was bringt uns nun die Nachiteration? Ganz einfach: Anstatt die komplexe Aufgabe des Lösen eines linearen Gleichungssystems in hoher Genauigkeit durchzuführen, tun wir das schneller⁴¹ 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 le-

⁴¹Es 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. Update 2013: Ähnliches gilt heute für GPU–Berechnungen auf der Grafikkarte und die dort verfügbaren Techniken.

```
%% VorElim2.m (Numerik 1)
%% -----
%% Vorwaartselimination 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.

ichte Abschwächung einer Aussage aus dem Buch von Higham [17]).

Satz 4.38 *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 relativen Genauigkeit*

$$\frac{\|\hat{x} - x\|_\infty}{\|x\|_\infty} \approx \begin{cases} \hat{u}, & \tilde{u} \sim \hat{u}^2, \\ 2(n + 1)\hat{u}\|A\|_\infty, & \tilde{u} \sim \hat{u}. \end{cases} \quad (4.39)$$

Wie man sowas beweist, zeigt Anhang A.2.

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.

Spezielle lineare Gleichungssysteme | 5

Bevor wir uns spezielle Systeme ansehen, erst eine kleine Variation von Satz (4.29) – der einzige Unterschied besteht darin, daß jetzt ebenfalls alle Diagonalelemente von U den Wert 1 haben.

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 $l_{jj} = u_{jj} = 1$, $d_{jj} \neq 0$, $j = 1, \dots, n$, und*

$$PA = LDU. \quad (5.1)$$

Beweis: Nach Satz (4.29) gibt es P, L, U' so daß $PA = LU'$ und $u'_{jj} \neq 0$, $j = 1, \dots, n$. Dann setzen wir

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

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*

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

*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“ unterscheiden⁴², oder aber die Begriffe „strikt positiv definit“ und „positiv definit“ verwenden⁴³. Deswegen empfiehlt 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

$$x^T A x \geq 0, \quad x \in \mathbb{C}^n$$

insbesondere bedeutet, daß $x^T A x$ eine reelle Zahl ist, und das ist wiederum nur dann der Fall, wenn A eine **hermitesche Matrix** ist, also wenn $A^H = A$ 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 die **Cholesky-Zerlegung**

$$A = G G^T \tag{5.2}$$

hat.

Beweis: Zuerst bemerken wir, daß A genau dann positiv definit ist, wenn alle Hauptuntermatrizen A_m , $m = 1, \dots, n$, 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

$$LDU = A = A^T = U^T D L^T$$

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

$$A = LDL^T$$

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

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

Damit ist

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

⁴²Was meistens in der deutschsprachigen Literatur geschieht.

⁴³Wie in der englischsprachigen Literatur. Richtig verwirrend wird es dann, wenn Deutsche in Englisch schreiben und ihre gewohnte Begriffswelt beibehalten.

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

Übung 5.1 Zeigen Sie, daß eine symmetrische Matrix $A \in \mathbb{R}^{n \times n}$ genau dann positiv definit ist, wenn alle Hauptminoren positiv sind: $\det A_m > 0$, $m = 1, \dots, n$. \diamond

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

$$a_{jk} = \sum_{r=1}^j g_{jr}g_{kr}, \quad 1 \leq j \leq k \leq n, \quad (5.3)$$

aus. Das liefert uns, daß

$$g_{jj} = \sqrt{a_{jj} - \sum_{r=1}^{j-1} g_{jr}^2}, \quad j = 1, \dots, n, \quad (5.4)$$

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

5.2 Bandierte Systeme

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

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

Der Speicheraufwand für eine (p, q) -bandierte Matrix ist nur $O((p + q + 1)n)^{44}$, 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.

⁴⁴Für Erbsenzähler: Es sind $n^2 - \frac{(n-p)(n-p-1)}{2} - \frac{(n-q)(n-q-1)}{2}$ Einträge

```
%% 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
end
%endfunction
```

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

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

$$A = \begin{bmatrix} \alpha & w^T \\ v & B \end{bmatrix}, \quad \alpha \neq 0, v, w \in \mathbb{R}^n, B \in \mathbb{R}^{n \times n},$$

wobei $v_{p+1} = \dots = v_n = 0$, $w_{q+1} = \dots = w_n = 0$, und B eine (p, q) -bandierte Matrix ist. Nach einem Schritt Gauß-Elimination ist dann

$$A = \begin{bmatrix} 1 & 0 \\ \frac{v}{\alpha} & I_n \end{bmatrix} \begin{bmatrix} \alpha & w^T \\ 0 & B - \frac{vw^T}{\alpha} \end{bmatrix}.$$

Da

$$vw^T = \begin{bmatrix} v_1 w_1 & \dots & v_1 w_q \\ \vdots & \ddots & \vdots \\ v_p w_1 & \dots & v_p w_q \\ & & & 0 \end{bmatrix}$$

ebenfalls (p, q) -bandiert ist, besitzt die (p, q) -bandierte Matrix $B - vw^T/\alpha$ eine $(p, 0)$ - bzw. $(0, q)$ -bandierte LU-Zerlegung

$$B - \frac{vw^T}{\alpha} = \tilde{L}\tilde{U},$$

und daher ist

$$A = \underbrace{\begin{bmatrix} 1 & 0 \\ v & \tilde{L} \end{bmatrix}}_{=:L} \underbrace{\begin{bmatrix} \alpha & w^T \\ 0 & \tilde{U} \end{bmatrix}}_{=:U},$$

wobei L eine $(p, 0)$ -bandierte Matrix und U eine $(0, q)$ -bandierte Matrix ist. \square

Eine *bandierte* Version der Gauß-Elimination findet sich in `BGauss.m`.

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 [22].

Definition 5.7 Für $A \in \mathbb{R}^{n \times n}$ und $I, J \subseteq \{1, \dots, n\}$ mit $\#I = \#J (= k)$ bezeichnet

$$A(I, J) = A \begin{pmatrix} i_1, \dots, i_k \\ j_1, \dots, j_k \end{pmatrix} = \begin{bmatrix} a_{i_1, j_1} & \dots & a_{i_1, j_k} \\ \vdots & \ddots & \vdots \\ a_{i_k, j_1} & \dots & a_{i_k, j_k} \end{bmatrix}, \quad \begin{matrix} i_1 < \dots < i_k, \\ j_1 < \dots < j_k, \end{matrix}$$

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

$$A = A(N, N).$$

```

%% 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 $O(n^2)$ solange $p, q \ll n$.

Definition 5.8 Eine Matrix $A \in \mathbb{R}^{n \times n}$ heißt **total positiv**, wenn

$$\det A(I, J) > 0, \quad I, J \subseteq \{1, \dots, n\}, \quad \#I = \#J, \quad (5.7)$$

und **total nichtnegativ**, wenn

$$\det A(I, J) \geq 0, \quad I, J \subseteq \{1, \dots, n\}, \quad \#I = \#J, \quad (5.8)$$

Bemerkung 5.9 Ist eine Matrix A total nichtnegativ, dann ist

$$a_{jk} \geq 0, \quad j, k = 1, \dots, n.$$

Versucht man, nichtnegative Matrizen zu „basteln“, dann stellt man schnell fest, daß das gar nicht so einfach ist. Es gibt sie allerdings wirklich, und zwar sogar von beliebiger Dimension.

Beispiel 5.10 Die n -te **Pascal-Matrix**

$$P_n := \left[\binom{j+k-2}{j-1} : j, k = 1, \dots, n \right], \quad n \in \mathbb{N},$$

ist 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

$$A = LU \quad \text{und} \quad L \geq 0, \quad U \geq 0. \quad (5.9)$$

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 9.6.

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 $m > 1$

$$0 \leq \det A(\{1, m\}, \{1, k\}) = \begin{vmatrix} a_{11} & a_{1k} \\ a_{m1} & a_{mk} \end{vmatrix} = -a_{1k}a_{m1} \quad \Rightarrow \quad a_{m1} = 0.$$

Das heißt, wenn $a_{11} = 0$ ist, dann ist entweder $a_{1k} = 0$, $k = 1, \dots, n$, oder $a_{k1} = 0$, $k = 1, \dots, n$, oder sogar beides, aber auf alle Fälle ist dann A nicht invertierbar. \square

Lemma 5.14 Sei $A \in \mathbb{R}^{n \times n}$ total nichtnegativ und invertierbar. Dann ist die Matrix \tilde{A} , die man nach einem Gauß–Eliminationsschritt als

$$A \rightarrow A^{(1)} := \begin{bmatrix} \mathbf{a}_{11} & \mathbf{v}^T \\ 0 & \tilde{A} \end{bmatrix}, \quad \tilde{A} = [\tilde{a}_{jk} : j, k = 2, \dots, n] \in \mathbb{R}^{n-1 \times n-1}.$$

erhält, auch total nichtnegativ und invertierbar.

Beweis: Für beliebige $I, J \subseteq \{2, \dots, n\}$ betrachten wir

$$\det \tilde{A}(I, J) = \frac{1}{a_{11}} \det A^{(1)}(I \cup \{1\}, J \cup \{1\}). \quad (5.10)$$

Mit

$$A = \begin{bmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_n^T \end{bmatrix} \quad \text{und} \quad \alpha_j = \frac{a_{j1}}{a_{11}}, \quad j = 2, \dots, n,$$

ist

$$A^{(1)} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T - \alpha_2 \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_n^T - \alpha_n \mathbf{a}_1^T \end{bmatrix}$$

und damit

$$A^{(1)}(I \cup \{1\}, J \cup \{1\}) = \begin{bmatrix} \mathbf{a}_1^T(J \cup \{1\}) \\ \mathbf{a}_{i_1}^T(J \cup \{1\}) - \alpha_{i_1} \mathbf{a}_1^T(J \cup \{1\}) \\ \vdots \\ \mathbf{a}_{i_k}^T(J \cup \{1\}) - \alpha_{i_k} \mathbf{a}_1^T(J \cup \{1\}) \end{bmatrix}. \quad (5.11)$$

Da die Determinante invariant unter Addition von Vielfachen von Zeilen ist, haben wir also, daß

$$\det A^{(1)}(I \cup \{1\}, J \cup \{1\}) = \begin{vmatrix} \mathbf{a}_1^T(J \cup \{1\}) \\ \mathbf{a}_{i_1}^T(J \cup \{1\}) \\ \vdots \\ \mathbf{a}_{i_k}^T(J \cup \{1\}) \end{vmatrix} = \det A(I \cup \{1\}, J \cup \{1\}),$$

und setzen wir dies in (5.10) ein, dann erhalten wir, daß

$$\det \tilde{A}(I, J) = \frac{1}{a_{11}} \det A(I \cup \{1\}, J \cup \{1\}), \quad I, J \subset \{2, \dots, n\}, \quad \#I = \#J. \quad (5.12)$$

Damit ist \tilde{A} total nichtnegativ und außerdem ist

$$\det \tilde{A} = \frac{1}{a_{11}} \det A > 0$$

und damit ist \tilde{A} auch invertierbar. \square

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

$$A^{(1)} = \begin{bmatrix} a_{11} & A(\{1\}, \{2, \dots, n\}) \\ 0 & \tilde{A} \end{bmatrix},$$

wenden die Induktionshypothese auf \tilde{A} an (das geht nach Lemma 5.14) und erhalten $\tilde{L} \geq 0$ und $\tilde{U} \geq 0$ so daß $\tilde{A} = \tilde{L}\tilde{U}$. Dann ist aber

$$A = \underbrace{\begin{bmatrix} 1 & 0 \\ a_{11}^{-1}A(\{2, \dots, n\}, 1) & \tilde{L} \end{bmatrix}}_{=:L} \underbrace{\begin{bmatrix} a_{11} & A(\{1\}, \{2, \dots, n\}) \\ 0 & \tilde{U} \end{bmatrix}}_{=:U},$$

und es ist $L, U \geq 0$. \square

Korollar 5.15 Die LU–Zerlegung aus (5.9) erhält man durch **Gauß–Elimination** ohne Pivotsuche (oder durch Doolittle).

Da nun

$$|\hat{L}| \quad |\hat{U}| \leq |L| \quad |U| + (|\hat{L} - L| + |\hat{U} - U|) = (1 + O(\hat{u}^2)) |A|,$$

ist auch

$$\| |\hat{L}| \quad |\hat{U}| \|_{\infty} = (1 + O(\hat{u}^2)) \|A\|_{\infty}$$

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

$$\frac{\|\hat{x} - x\|_{\infty}}{\|x\|_{\infty}} \leq \frac{8n\hat{u}}{1 - \hat{u}\kappa_{\infty}(A)} \kappa_{\infty}(A). \quad (5.13)$$

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 $L, U \in \mathbb{R}^{n \times n}$, $\ell_{jj} = 1$, $u_{jj} \neq 0$, $j = 1, \dots, n$, gibt, so daß $A = LU$.

There are things that are facts, in a statistical sense, on paper, on a tape recorder, in evidence. And there are things that are facts because they have to be facts, because nothing makes any sense otherwise.

Raymond Chandler, „Payback“

Iterative Verfahren für lineare Gleichungssysteme

6

Wir betrachten wieder eine invertierbare Matrix $A \in \mathbb{R}^{n \times n}$ und ein Gleichungssystem

$$Ax = b. \quad (6.1)$$

Ein **iteratives Verfahren** besteht aus einer Berechnungsvorschrift

$$x^{(j+1)} = F(x^{(j)})$$

mit dem Ziel, daß $x^{(j)} \rightarrow x$ für $j \rightarrow \infty$, wobei natürlich x die Lösung von (6.1) sein soll. Im Gegensatz zu den direkten Verfahren berechnen wir also keine „exakte“ Lösung⁴⁵ mehr, sondern versuchen, uns näher und näher an die Lösung „heranzutasten“.

Zuerst ein ganz billiger Taschenspielertrick, um das Problem ein klein wenig umzuformen: Ist x die Lösung von $Ax = b$, so gilt für beliebiges invertierbares $B \in \mathbb{R}^{n \times n}$

$$b = Ax = Bx + (A - B)x \quad \Leftrightarrow \quad x = B^{-1}b + (I - B^{-1}A)x,$$

also ist x ein **Fixpunkt** von

$$F(x) = B^{-1}b + (I - B^{-1}A)x. \quad (6.2)$$

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.

⁴⁵Bei exakter Rechnung (z.B. in Computeralgebrasystemen) würden die direkten Verfahren ja nach endlich vielen Schritten eine exakte Lösung liefern.

Definition 6.1 Eine Abbildung $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ heißt **Kontraktion** bezüglich einer Norm $\|\cdot\|$, wenn es eine Konstante $\rho < 1$ gibt, so daß

$$\|F(x) - F(y)\| \leq \rho \|x - y\|, \quad x, y \in \mathbb{R}^n. \quad (6.3)$$

Ein Punkt $x \in \mathbb{R}^n$ heißt **Fixpunkt** einer Abbildung $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$, wenn $F(x) = x$ ist.

Satz 6.2 (Banachscher Fixpunktsatz für den \mathbb{R}^n) Ist $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ eine Kontraktion, dann hat F genau einen Fixpunkt x^* und für jedes $x^{(0)}$ konvergiert die Folge

$$x^{(j+1)} = F(x^{(j)}), \quad j \in \mathbb{N}_0, \quad (6.4)$$

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

$$\|x^{(j+1)} - x^{(j)}\| = \|F(x^{(j)}) - F(x^{(j-1)})\| \leq \rho \|x^{(j)} - x^{(j-1)}\| \leq \dots \leq \rho^j \|x^{(1)} - x^{(0)}\|$$

und daher, für $N > 0$,

$$\begin{aligned} \|x^{(j+N)} - x^{(j)}\| &\leq \sum_{k=1}^N \|x^{(j+k)} - x^{(j+k-1)}\| \leq \sum_{k=0}^{N-1} \rho^{j+k} \|x^{(1)} - x^{(0)}\| \\ &\leq \|x^{(1)} - x^{(0)}\| \rho^j \sum_{k=0}^{\infty} \rho^k = \frac{\rho^j}{1-\rho} \|x^{(1)} - x^{(0)}\|. \end{aligned}$$

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

$$\|F(x^*) - x^*\| \leq \|F(x^*) - F(x^{(j)})\| + \|x^* - x^{(j+1)}\| \leq \rho \|x^* - x^{(j)}\| + \|x^* - x^{(j+1)}\|,$$

gilt und da die rechte Seite für $j \rightarrow \infty$ gegen 0 konvergiert, ist $F(x^*) = x^*$. \square

Jetzt aber zurück zu (6.2). Das daraus entstehende Lösungsverfahren wird die Iteration

$$x^{(j+1)} = F(x^{(j)}) = B^{-1}b + (I - B^{-1}A)x^{(j)} \quad (6.5)$$

verwenden, und 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 Matrixklassen A (und passendem B), so daß (6.5) garantiert konvergiert.

Bemerkung 6.3 Wenn die Iteration (6.5) gegen x^* konvergiert, dann ist

$$x^* = B^{-1}b + (I - B^{-1}A)x^*.$$

Dies gilt sogar für jede lineare Iteration der Form

$$x^{(j+1)} = F(x^{(j)}) = Gx^{(j)} + g$$

mit $G \in \mathbb{R}^{n \times n}$ und $g \in \mathbb{R}^n$. Schließlich ist ja für jedes $j \in \mathbb{N}$

$$\begin{aligned} \|x^* - F(x^*)\| &\leq \|x^* - x^{(j+1)}\| + \|F(x^*) - x^{(j+1)}\| \\ &= \|x^* - x^{(j+1)}\| + \|F(x^* - x^{(j)})\| \leq \|x^* - x^{(j+1)}\| + \|G\| \|x^* - x^{(j)}\| \end{aligned}$$

und da die rechte Seite für $j \rightarrow \infty$ gegen Null konvergiert während die linke Seite von j unabhängig ist, muß $x^* = F(x^*)$ sein.

6.1 Der Spektralradius und ein allgemeines Konvergenzkriterium

Wir kommen zum entscheidenden Begriff für die Konvergenz des iterativen Verfahrens.

Definition 6.4 Das *Spektrum* $S(A)$ einer Matrix $A \in \mathbb{R}^{n \times n}$ ist die Menge⁴⁶

$$S(A) = \{\lambda \in \mathbb{C} : \ker_{\mathbb{C}^n}(A - \lambda I) \neq \{0\}\}$$

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

$$\rho(A) = \limsup_{j \rightarrow \infty} \|A^j\|^{1/j} \quad (6.6)$$

und erfüllt somit $\rho(A) \leq \|A\|$.

⁴⁶Zur Erinnerung: Auch wenn die Matrix reell ist, können ihre Eigenwerte sehr wohl komplex sein, daher ist das Spektrum in Normalfall eine Teilmenge der komplexen Ebene.

Beweis: Wir erweitern das Problem auf \mathbb{C}^n , es sei also jetzt $\|\cdot\|$ eine Fortsetzung der Vektornorm auf \mathbb{C}^n ⁴⁷.

Sei $\lambda \in S(A)$ und $x \in \mathbb{C}^n$ ein dazugehöriger Eigenvektor mit $\|x\| = 1$. Dann ist

$$\|A^j\| \geq \|A^j x\| = |\lambda|^j \|x\| = |\lambda|^j$$

und damit ist

$$\limsup_{j \rightarrow \infty} \|A^j\|^{1/j} \geq \rho(A).$$

Für die Umkehrung gibt es eine invertierbare Matrix $S \in \mathbb{C}^{n \times n}$, die A in ihre **Jordan-Normalform**

$$S^{-1}AS =: J = \begin{bmatrix} \lambda_1 & * & & \\ & \ddots & \ddots & \\ & & \lambda_{n-1} & * \\ & & & \lambda_n \end{bmatrix}, \quad |\lambda_1| \leq \dots \leq |\lambda_n|, \quad * \in \{0, 1\}.$$

überführt. Für jedes $\varepsilon > 0$ sei außerdem

$$D_\varepsilon = \begin{bmatrix} 1 & & & \\ & \varepsilon & & \\ & & \ddots & \\ & & & \varepsilon^{n-1} \end{bmatrix},$$

dann ist

$$D_\varepsilon J D_\varepsilon^{-1} = \begin{bmatrix} \lambda_1 & * & & \\ & \ddots & \ddots & \\ & & \lambda_{n-1} & * \\ & & & \lambda_n \end{bmatrix}, \quad * \in \{0, \varepsilon\}$$

und daher

$$\|D_\varepsilon J D_\varepsilon^{-1}\| = |\lambda_n| + f(\varepsilon), \quad f \in C(\mathbb{R}), f(0) = 0.$$

Somit ist, für festes $\varepsilon > 0$,

$$\begin{aligned} \|A^j\| &\leq \|S\| \left\| (S^{-1}AS)^j \right\| \|S^{-1}\| = \kappa(S) \|J^j\| \leq \kappa(S) \kappa(D_\varepsilon) \left\| (D_\varepsilon J D_\varepsilon^{-1})^j \right\| \\ &\leq \kappa(S) \kappa(D_\varepsilon) \left\| D_\varepsilon J D_\varepsilon^{-1} \right\|^j = \kappa(S) \kappa(D_\varepsilon) (|\lambda_n| + f(\varepsilon))^j, \end{aligned}$$

also ist

$$\limsup_{j \rightarrow \infty} \|A^j\|^{1/j} \leq \limsup_{j \rightarrow \infty} \underbrace{(\kappa(S) \kappa(D_\varepsilon))^{1/j}}_{=1} (|\lambda_n| + f(\varepsilon)),$$

⁴⁷Die p -Normen lassen sich beispielsweise ohne Änderung auf \mathbb{C}^n übertragen, nur spielt der Betrag jetzt eine etwas wesentlichere Rolle.

und damit

$$\limsup_{j \rightarrow \infty} \left\| A^j \right\|^{1/j} \leq \max_{\lambda \in S(A)} |\lambda| + f(\varepsilon),$$

was mit $\varepsilon \rightarrow 0$ den Beweis vervollständigt. \square

Bemerkung 6.6 (Spektralradius)

1. Es ist schon wichtig, komplexe Eigenwerte mitzubersichtigen: 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 (6.6) erhält, da $A^2 = -I$ ist.

2. Der Vorteil von (6.6) besteht darin, daß wir hiermit den Spektralradius einer Matrix mit „reellen Mitteln“ bestimmen können.
3. Außerdem ist (6.6) 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 (6.5) konvergiert genau dann für jeden Startvektor $x^{(0)} \in \mathbb{C}^n$, wenn

$$\rho(I - B^{-1}A) < 1. \quad (6.7)$$

Beweis: Ist (6.7) erfüllt, dann gibt es einen Index $m > 0$, so daß $\|(I - B^{-1}A)^m\| < 1$, das heißt, die Abbildung

$$F(x) := (I - B^{-1}A)^m x$$

ist eine Kontraktion und konvergiert, nach Satz 6.2 gegen einen eindeutigen Fixpunkt x^* , d.h., für $j = 0, \dots, m-1$ gilt

$$F^k(x^{(j)}) = x^{(j+km)} \rightarrow x^*,$$

also konvergiert auch die ganze Folge.

Nehmen wir umgekehrt an, daß das Iterationsverfahren (6.5) 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

$$x^{(1)} - x^* = (I - B^{-1}A)x^{(0)} + B^{-1}b - (I - B^{-1}A)x^* - B^{-1}b = (I - B^{-1}A)y = \lambda y,$$

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 (6.5) konvergiert für jeden Startvektor $x^{(0)} \in \mathbb{R}^n$ wenn

1. $\rho(I - B^{-1}A) < 1$.
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 $\rho(I - B^{-1}A) < 1$ und es sei x die Lösung von $Ax = b$. Dann gilt, für jeden Startvektor $x^{(0)}$,

$$\limsup_{j \rightarrow \infty} \left(\frac{\|x^{(j)} - x\|}{\|x^{(0)} - x\|} \right)^{1/j} \leq \rho(I - B^{-1}A). \quad (6.8)$$

Beweis: Nach Proposition 6.5 gibt es für alle $\varepsilon \in (0, 1 - \rho(I - B^{-1}A))$ einen Index $j_0 \in \mathbb{N}$ und eine Konstante $C > 0$, so daß

$$\left\| (I - B^{-1}A)^j \right\| \leq C \left(\varepsilon + \rho(I - B^{-1}A) \right)^j < 1, \quad j > j_0,$$

und somit ist

$$\begin{aligned} \|x^{(j)} - x\| &= \left\| (I - B^{-1}A)^j (x^{(0)} - x) \right\| \leq \left\| (I - B^{-1}A)^j \right\| \|x^{(0)} - x\| \\ &\leq C \left(\varepsilon + \rho(I - B^{-1}A) \right)^j \|x^{(0)} - x\|, \quad j > j_0, \end{aligned}$$

also

$$\limsup_{j \rightarrow \infty} \left(\frac{\|x^{(j)} - x\|}{\|x^{(0)} - x\|} \right)^{1/j} \leq \varepsilon + \rho(I - B^{-1}A)$$

und da ε beliebig war, folgt (6.8) □

Die **Konvergenzordnung**, die (6.8) suggeriert, ist

$$\|x^{(j)} - x\| \leq C\rho^j. \quad \rho = \rho(I - B^{-1}A) < 1,$$

der Fehler fällt also exponentiell. Ist beispielsweise $\rho = 1/10$, dann hätte man also die Fehler $1/10$, $1/100$, $1/1000$ und so weiter, in jedem Iterationsschritt wird also eine Dezimalstelle des Ergebnisses korrekt. Aus diesem Grund wird dieses Konvergenzverhalten auch als **lineare Konvergenz** bezeichnet.

6.2 Gauß–Seidel–Iteration und Jacobi–Iteration

Erinnern wir uns an unsere Anforderungen an die Matrix B für die *implizite* Iteration

$$Bx^{(j+1)} = b - (A - B)x^{(j)},$$

also Invertierbarkeit und einfache Bestimmbarkeit aus A . Der einfachste Fall ist sicherlich

$$B = \text{diag} [a_{jj} : j = 1, \dots, n],$$

vorausgesetzt natürlich, daß $a_{jj} \neq 0$, $j = 1, \dots, n$. Und das ist auch bereits das **Jacobi-Verfahren**, in der deutschsprachigen Literatur auch gerne als **Gesamtschrittverfahren** bezeichnet:

$$\begin{bmatrix} a_{11} & & & \\ & \ddots & & \\ & & a_{nn} & \\ & & & \end{bmatrix} x^{(j+1)} = b - \begin{bmatrix} 0 & a_{12} & \dots & a_{1n} \\ a_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n} \\ a_{n1} & \dots & a_{n,n-1} & 0 \end{bmatrix} x^{(j)} \quad (6.9)$$

beziehungsweise

$$x_k^{(j+1)} = \left(b_k - \sum_{r \neq k} a_{kr} x_r^{(j)} \right) / a_{kk}, \quad k = 1, \dots, n. \quad (6.10)$$

Bei der **Gauß–Seidel–Iteration**, dem **Einzelschrittverfahren**, setzt man

$$B = \begin{bmatrix} a_{11} & & 0 \\ \vdots & \ddots & \\ a_{n1} & \dots & a_{nn} \end{bmatrix},$$

also

$$\begin{bmatrix} a_{11} & & 0 \\ \vdots & \ddots & \\ a_{n1} & \dots & a_{nn} \end{bmatrix} x^{(j+1)} = b - \begin{bmatrix} 0 & a_{12} & \dots & a_{1n} \\ & \ddots & \ddots & \vdots \\ & & \ddots & a_{n-1,n} \\ & & & 0 \end{bmatrix} x^{(j)}, \quad (6.11)$$

beziehungsweise

$$x_k^{(j+1)} = \left(b_k - \sum_{r=1}^{k-1} a_{kr} x_r^{(j+1)} - \sum_{r=k+1}^n a_{kr} x_r^{(j)} \right) / a_{kk}, \quad k = 1, \dots, n. \quad (6.12)$$

Es gibt noch eine andere Interpretation von (6.12), 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?

```

%% JacobiIt.m (Numerik 1)
%% -----
%% Jacobi-Iterationsschritt
%% Eingabe:
%%   A      Matrix
%%   x      Naehierung 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 (6.10).

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 Matrizen⁴⁸ 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.*

⁴⁸Also solchen mit kleinem Spektralradius

```
%% 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    Naehierung 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 (6.12).

```

%% 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.

Beweis: Wir verwenden die Zerlegung

$$A = \underbrace{L + D}_{=B} + \underbrace{L^T}_{=A-B},$$

wobei L eine *strikte* untere Dreiecksmatrix ist, das heißt, $\ell_{jj} = 0$, $j = 1, \dots, n$. Wir müssen nun also zeigen, daß

$$\rho(G) := \rho\left(B^{-1}(B - A)\right) = \rho\left(-(L + D)^{-1} L^T\right) < 1. \quad (6.13)$$

Wegen der positiven Definitheit von A ist $d_{jj} = a_{jj} > 0$, $j = 1, \dots, n$ ⁴⁹, und damit ist die Matrix G_1 , definiert durch

$$\begin{aligned} G_1 &:= D^{1/2} G D^{-1/2} = -D^{1/2} (L + D)^{-1} L^T D^{-1/2} \\ &= -D^{1/2} (L + D)^{-1} D^{1/2} D^{-1/2} L^T D^{-1/2} \\ &= -\left(D^{-1/2} (L + D) D^{-1/2}\right)^{-1} \underbrace{\left(D^{-1/2} L D^{-1/2}\right)^T}_{=:L_1} = -(I + L_1)^{-1} L_1^T \end{aligned}$$

wohldefiniert und *ähnlich* zu G und hat deswegen auch dieselben Eigenwerte wie G ⁵⁰. Sei nun $x \in \mathbb{C}^n$, $x^H x = 1$ ⁵¹, ein Eigenvektor von G_1 zum Eigenwert λ , dann heißt dies, daß

$$-L_1^T x = \lambda (I + L_1) x$$

und somit

$$-\underbrace{x^H L_1^T x}_{=(L_1 x)^H x = (x^H L_1 x)^H = \overline{x^H L_1 x}} = \lambda \left(1 + x^H L_1 x\right).$$

Schreiben wir $x^H L_1 x = \alpha + i\beta$, dann ist

$$|\lambda|^2 = \left| \frac{-\alpha + i\beta}{1 + \alpha + i\beta} \right|^2 = \frac{\alpha^2 + \beta^2}{1 + 2\alpha + \alpha^2 + \beta^2}. \quad (6.14)$$

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 definit⁵², 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 (6.14) eingesetzt, $|\lambda| < 1$ liefert. □

⁴⁹Sonst wäre $e_j^T A e_j \leq 0$ für ein j .

⁵⁰Ist y ein Eigenvektor von G zum Eigenwert λ , dann ist $D^{1/2} y$ ein Eigenvektor von G_1 zum Eigenwert λ und umgekehrt.

⁵¹ $x^H = [\bar{x}_j : j = 1, \dots, n]^T$

⁵²Zur Erinnerung: Bei einer *komplexen* Matrix gehört es zur Definition von positiver Definitheit, daß die Matrix hermitesch ist. Bei reellen Matrizen fordert man es oder eben auch nicht ...

Definition 6.11 Eine Matrix $A \in \mathbb{R}^{n \times n}$ heißt **zeilenweise diagonaldominant**, wenn die Diagonalelemente den „Rest“ der Zeile im Absolutbetrag dominieren, wenn also

$$a_{jj} \geq \sum_{k \neq j} |a_{jk}|, \quad j = 1, \dots, n,$$

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 ($B = \text{diag } A$) gilt

$$\|I - B^{-1}A\|_{\infty} = \max_{j=1, \dots, n} \frac{1}{|a_{jj}|} \sum_{k \neq j} |a_{jk}|$$

und im Falle strikt diagonaldominanter Matrizen ist somit

$$\rho(I - B^{-1}A) \leq \|I - B^{-1}A\|_{\infty} < 1.$$

Für Gauß–Seidel müssen wir uns ein bißchen mehr anstrengen. Hierzu schreiben wir

$$A = L + D + U, \quad \ell_{jj} = u_{jj} = 0,$$

wobei $d_{jj} \neq 0$ ist, und setzen $L_1 = -D^{-1}L$, $U_1 = -D^{-1}U$. Die zu betrachtende Matrix ist

$$\begin{aligned} I - B^{-1}A &= I - (L + D)^{-1}(L + D + U) = I - (-DL_1 + D)^{-1}(L + D + U) \\ &= I - (I - L_1)^{-1}D^{-1}(L + D + U) = I - (I - L_1)^{-1}(I - L_1 - U_1) \\ &= I - I + (I - L_1)^{-1}U_1 = (I - L_1)^{-1}U_1 =: H, \end{aligned}$$

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

$$|U_1| 1_n = (|J| - |L_1|) 1_n \leq (\|J\|_{\infty} I - |L_1|) 1_n. \quad (6.15)$$

L_1 und $|L_1|$ sind strikte untere Dreiecksmatrizen und damit **nilpotent**, das heißt,

$$L_1^n = |L_1|^n = 0 \quad \Rightarrow \quad \begin{cases} (I - L_1)^{-1} = I + L_1 + \dots + L_1^{n-1} \\ (I - |L_1|)^{-1} = I + |L_1| + \dots + |L_1|^{n-1} \end{cases}$$

und damit ist

$$\left| (I - L_1)^{-1} \right| = \left| I + L_1 + \dots + L_1^{n-1} \right| \leq I + |L_1| + \dots + |L_1|^{n-1} = (I - |L_1|)^{-1}. \quad (6.16)$$

Mit (6.15) und (6.16) ist demnach

$$\begin{aligned}
 |H| \mathbf{1}_n &\leq \left| (I - L_1)^{-1} \right| |U_1| \mathbf{1}_n \leq (I - |L_1|)^{-1} (\|J\|_\infty I - |L_1|) \mathbf{1}_n \\
 &= (I - |L_1|)^{-1} (I - |L_1| + (\|J\|_\infty - 1)I) \mathbf{1}_n \\
 &= \mathbf{1}_n + \underbrace{(\|J\|_\infty - 1)}_{<0} \underbrace{(I - |L_1|)^{-1}}_{\geq I} \mathbf{1}_n \\
 &\leq (I + (\|J\|_\infty - 1)I) \mathbf{1}_n \leq \|J\|_\infty \mathbf{1}_n < \mathbf{1}_n.
 \end{aligned}$$

Also ist $|H| \mathbf{1}_n < \mathbf{1}_n$, also alle Zeilensummen von $|H|$ kleiner als 1 und damit ist $\|H\|_\infty < 1$, also auch $\rho(H) < 1$. □

Für den Ausgleichsanspruch nach §1587g Abs. 1 Satz 1 gelten die §§ 1580, 1585 Abs. 1 Satz 2,3 und §1585b Abs. 2,3 entsprechend.

BGB, §1587k. Abs. 1

Ausgleichsrechnung | 7

Bisher haben wir uns mit *quadratischen* linearen Gleichungssystemen der Form $Ax = b$, $A \in \mathbb{R}^{n \times n}$ und deren numerischer Lösung beschäftigt, das heißt, die Anzahl der Gleichung und die Anzahl der Unbekannten stimmen überein – eine notwendige Voraussetzung wenn man für *alle* rechten Seiten b eine *eindeutige* Lösung erhalten will. Diese Theorie läßt sich relativ einfach auf *unterbestimmte* Gleichungssysteme erweitern, bei denen man weniger Gleichungen als Unbekannte hat. Man verliert zwar die Eindeutigkeit der Lösung, aber solange die Matrix vollen Rang hat, kann man sich (zumindest formal) auf ein eindeutig lösbares *Teilsystem* zurückziehen⁵³. Besser macht man es aber auf andere Art und Weise

7.1 Inverse Probleme – ein kurzer Exkurs

Nehmen wir jetzt also an, wir hätten es mit einem *unterbestimmten* linearen Gleichungssystem

$$Ax = b, \quad A \in \mathbb{R}^{m \times n}, \quad m < n, \quad b \in \mathbb{R}^m, \quad (7.1)$$

zu tun, dessen Matrix A , der Einfachheit halber, aber den *maximalen Rang* m hat⁵⁴. Anstatt nun aber den Lösungsraum einfach einzuschränken, suchen wir uns in unserer Lösungsmenge eine *beste* Lösung, wobei natürlich noch klarstellen müssen, in welchem Sinne die Lösung optimal zu sein hat.

Bemerkung 7.1 *Unterbestimmte Gleichungssysteme kommen in der Anwendungsrealität oft vor und zwar dann, wenn es um inverse Systeme geht. Dabei kennt man eine (hoffentlich oder näherungsweise) lineare Beziehung der Form $y = Ax$ zwischen den (gesuchten) Regelgrößen x und den gemessenen Werten y , nur sind es oftmals viel zu viele Regelgrößen und viel zu*

⁵³Die Auswahl der "besten" Lösung aus dem affinen Lösungsraum ist hingegen ein durchaus nicht-triviales Problem.

⁵⁴Das ist ohnehin der "generische" Fall, die Menge der Matrizen mit Rangdefekt bildet einen Nullmenge in der Menger aller $m \times n$ -Matrizen.

wenige Messungen⁵⁵. Das führt dann zu einem unterbestimmten System, aus dem x nun in irgendeiner Form bestimmt werden muss.

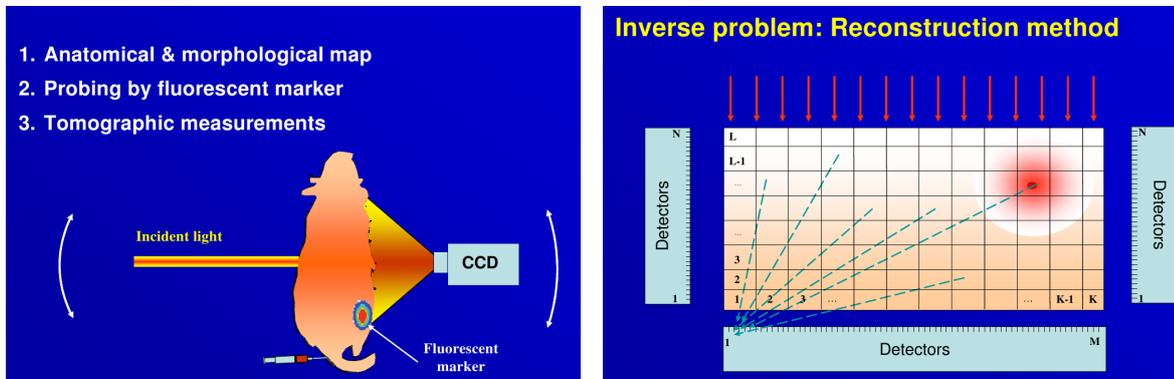


Abbildung 7.1: Optische Tomographie (*links*). Ein fluoreszierender Marker *innerhalb* eines Lebewesens (hier eine Maus) wird durch Laserbestrahlung aktiviert und das Streulicht am anderen Ende aufgelesen. Aus diesen Informationen soll die Lage des Markers rekonstruiert werden.

Im physikalischen Modell (*rechts*) wird der Lichtaustausch zwischen benachbarten Zellen durch entsprechende Beugungs-, Reflektions- und Absorptionsgesetze ermittelt, woraus sich der Lichteinfall auf den Detektoren ergibt.

Beispiel 7.2 (Optische Tomographie) In der optischen Tomographie werden Objekte, normalerweise Lebewesen⁵⁶ ausschließlich mit Hilfe von sichtbarem Licht durchleuchtet, wobei im Inneren ein fluoreszierender Marker platziert wird, siehe Abb. 7.1, der durch das einfallende (Laser-) Licht aktiviert wird. Es klingt ein wenig abwegig mit Hilfe von Licht durch eine Maus "hindurchschauen" zu wollen, aber tatsächlich kommt wirklich einiges an Licht auf der anderen Seite des Objekts an, allerdings durch die Moleküle⁵⁷ gestreut und gebrochen.

Zur Modellierung dieses Problems teilt man jeden Schnitt durch das Objekt und seine Umgebung in kleine Quadrate ein und modelliert den Lichtaustausch zwischen diesen Quadraten⁵⁸ sowie den Randquadraten und den Detektoren. Bezeichnet $x \in \mathbb{R}^{MN}$ die Lichtintensität an den Quadraten des $M \times N$ -Gitters und $y \in \mathbb{R}^K$ die gemessenen Werte an den K Detektoren, die um das Objekt herum oder hinter dem Objekt angebracht sind, dann erhalten wir ein Gleichungssystem der Form

$$y = Ax, \quad A \in \mathbb{R}^{K \times MN}.$$

⁵⁵Die Anzahl der Messungen kann aus vielerlei Gründen beschränkt sein: Kosten, technische Randbedingungen, Zeitbegrenzung ...

⁵⁶Und in diesem Sinne eigentlich eher Subjekte als Objekte.

⁵⁷Bekanntlicherweise bei Lebewesen zumeist Wasser.

⁵⁸Der Einfachheit halber werden hier normalerweise Quadrate und deren Nachbarn betrachtet.

Ab einer bestimmten Rekonstruktionsgenauigkeit hat man dann aber mit Sicherheit wesentlich mehr Quadrate als Detektoren und das Problem wird massiv unterbestimmt!

Jetzt aber genug der Realität und zurück zur Theorie. Da unser unterbestimmtes Gleichungssystem $Ax = b$ aus (7.1) normalerweise jede Menge Lösungen hat, genauer gesagt, einen $n - m$ -dimensionalen Lösungsraum, müssen wir uns aus diesen Lösungen eine passende Lösung aussuchen – warum also nicht gleich die Beste? Dazu lösen wir das **Optimierungsproblem**

$$\min_x \gamma(x), \quad Ax = b, \quad (7.2)$$

mit einem vorgegeben **Gütefunktional** $\gamma : \mathbb{R}^n \rightarrow \mathbb{R}$, das zumindest nach unten beschränkt sein sollte⁵⁹; diese untere Schranke können wir dann auch gleich auf Null setzen und so $\gamma : \mathbb{R}^n \rightarrow \mathbb{R}_+$ annehmen. Beliebte Werte für γ sind $\gamma(x) = \|x\|_p$, $1 \leq p \leq \infty$, oder

$$\gamma(x) = \frac{1}{2} x^T B x, \quad B \text{ positiv definit.} \quad (7.3)$$

Die Lösung von (7.3) wollen wir uns noch schnell etwas genauer ansehen. Dazu erinnern wir uns, daß ein *notwendiges* Kriterium für die Existenz eines Extremums von $f : \mathbb{R}^n \rightarrow \mathbb{R}$ unter der Nebenbedingung $g(x) = 0$, $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$, durch die Existenz **Lagrange-Multiplikatoren** $\lambda \in \mathbb{R}^m$ gegeben ist, daß also

$$\nabla f(x) - \nabla g(x) \lambda = 0, \quad \nabla g := \left[\frac{\partial}{\partial x_j} g_k : \begin{array}{l} j = 1, \dots, n \\ k = 1, \dots, m \end{array} \right], \quad (7.4)$$

gelten muss. Dieses Resultat lernt man entweder in der Analysis unter dem Stichwort "Extrema unter Nebenbedingungen", siehe z.B. [16], oder in der Optimierung, siehe z.B. [35, 40]. Mit $f(x) = \frac{1}{2} x^T B x$ und $g(x) = Ax - b$ ist nun ganz einfach $\nabla f(x) = Bx$ sowie $\nabla g(x) = A^T$, so daß unsere gesuchte Optimallösung als Lösung von

$$\begin{array}{l} Bx - A^T \lambda = 0 \\ Ax = b \end{array} \Leftrightarrow \begin{bmatrix} B & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ -\lambda \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}$$

gegeben ist. Daß dieses Extremum ein Minimum sein muss, liegt an der Tatsache, daß $x \mapsto x^T B x$ ein nach oben geöffnetes Paraboloid darstellt und deswegen nur ein Minimum haben kann, siehe auch den Beweis von Proposition 7.6. Anders gesagt: Die Minimallösung bestimmt man durch Lösen des *symmetrischen* quadratischen $m + n \times m + n$ -Systems

$$Cx = d, \quad C = \begin{bmatrix} B & A^T \\ A & 0 \end{bmatrix}, \quad d = \begin{bmatrix} 0 \\ b \end{bmatrix}, \quad (7.5)$$

wobei man sich nur um die ersten n Variablen zu kümmern braucht, die Werte von λ interessieren ja herzlich wenig.

Übung 7.1 Zeigen Sie: Hat A den maximalen Rang m , dann ist C aus (7.5) invertierbar.

◇

⁵⁹Sonst gibt es normalerweise kein Minimum!

7.2 Nun aber überbestimmt

Jetzt geht es aber um *überbestimmte* lineare Systeme der Form

$$Ax \simeq b, \quad A \in \mathbb{R}^{m \times n}, \quad m > n, \quad b \in \mathbb{R}^m, \quad (7.6)$$

bei denen $x \in \mathbb{R}^n$ gesucht wird. Im allgemeinen⁶⁰ wird das „Gleichungssystem“ (7.6) nicht mehr lösbar sein⁶¹, weswegen man das folgende Ziel verfolgt:

Man finde $x \in \mathbb{R}^n$, so daß $Ax - b$ möglichst klein wird.

„Möglichst klein“ kann beispielsweise bedeuten, daß man x so bestimmt, daß $\|Ax - b\|$ für eine vorgegebene Vektornorm minimiert wird – dabei wird das optimale x im allgemeinen sehr wohl von der gewählten Vektornorm abhängen.

Beispiel 7.3 Für

$$A = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}, \quad b_1 \leq b_2 \leq b_3$$

haben die Minimallösungen $x \in \mathbb{R}$ für $\|Ax - b\|_p$ die folgende Gestalt:

$p = 1$: Man sieht leicht, daß $b_1 \leq x \leq b_3$ gelten muß, und dann ist

$$\|Ax - b\|_1 = (x - b_1) + |x - b_2| + (b_3 - x) = b_3 - b_1 + |x - b_2|,$$

was für $x = b_2$ minimal wird.

$p = 2$: Differenzieren von

$$\|Ax - b\|_2^2 = 3x^2 - 2(b_1 + b_2 + b_3)x + b_1^2 + b_2^2 + b_3^2$$

liefert sofort, daß $x = (b_1 + b_2 + b_3) / 3$.

$p = \infty$: Da das Maximum von $|x - b_j|$ immer entweder für $j = 1$ oder $j = 3$ angenommen wird, ist die optimale Wahl, x von beiden gleich weit entfernt sein zu lassen, also $x = (b_3 + b_1) / 2$.

Im Spezialfall $b_1 = b_2 = b_3 =: b$ ergibt sich natürlich immer $x = b$, die Lösung des Gleichungssystems.

⁶⁰Das hängt natürlich von der rechten Seite b ab.

⁶¹Daher auch das Symbol „ \simeq “ in (7.6).

Übung 7.2 Zeigen Sie: Die Summe

$$\sum_{j=1}^n |x - b_j|, \quad b_1 \leq \dots \leq b_n,$$

wird durch den **Median** minimiert, also

$$x = \begin{cases} b_{m+1}, & n = 2m + 1, \\ \frac{1}{2}(b_m + b_{m+1}), & n = 2m. \end{cases}$$

Wann ist die Lösung eindeutig? ◇

In diesem Kapitel soll aber nur die Norm

$$\|\cdot\|_2 : x \mapsto \sqrt{x^T x}$$

betrachtet werden, was vor allem auch daran liegt, daß die Theorie am einfachsten ist. Neuerdings haben sich Lösungen in der 1-Norm in vielerlei Anwendungen als besser erwiesen, oftmals sind sie einfach robuster gegen Ausreißer, siehe auch Übung 7.2.

7.3 Least-squares-Lösungen und die Normalgleichungen

Definition 7.4 Ein Vektor $x \in \mathbb{R}^n$ heißt **Least-Squares-Lösung** von (7.6), wenn

$$\|Ax - b\|_2 \leq \|Ay - b\|_2, \quad y \in \mathbb{R}^n.$$

Bemerkung 7.5 Existiert eine Lösung des Gleichungssystems (7.6), dann ist diese natürlich auch Least-Squares-Lösung.

Das schöne an der $\|\cdot\|_2$ -Norm ist die Tatsache, daß wir die Least-Squares-Lösung einfach bestimmen können.

Proposition 7.6 Sei $A \in \mathbb{R}^{m \times n}$, $m \geq n$. Dann ist jede Lösung x der Normalgleichungen

$$\underbrace{A^T A}_{\in \mathbb{R}^{n \times n}} x = \underbrace{A^T b}_{\in \mathbb{R}^n} \quad (7.7)$$

eine Lösung des Least-Squares-Problems (7.6) und umgekehrt.

Korollar 7.7 Hat die Matrix A vollen Rang, dann ist die Lösung des Least-Squares-Problems eindeutig.

Beweis: Hat A vollen Rang n , dann hat auch $A^T A \in \mathbb{R}^{n \times n}$ den Rang n und ist damit invertierbar. Damit gibt es aber auch (nur) genau eine Lösung der Normalgleichungen. □

Beweis von Proposition 7.6: Wir betrachten das (quadratische) Funktional

$$\begin{aligned}\|Ax - b\|_2^2 &= (Ax - b)^T (Ax - b) = x^T A^T Ax - x^T A^T b - b^T Ax + b^T b \\ &= x^T A^T Ax - 2x^T A^T b + b^T b,\end{aligned}$$

das genau dann durch x minimiert wird, wenn die Norm durch x minimiert wird. Ist nun x ein Minimalstelle, so muss

$$0 = \nabla \|Ax - b\|_2^2 = 2A^T Ax - 2A^T b$$

sein, siehe Übung 7.3, also hat jede Least-Squares-Lösung x sich an die Normalgleichungen (7.7) zu halten.

Erfüllt umgekehrt x (7.7), dann erhalten wir für jedes $y \in \mathbb{R}^n$, daß

$$\begin{aligned}\|Ay - b\|_2^2 &= \|Ax - b\|_2^2 + \underbrace{(y - x)^T A^T A (y - x)}_{\geq 0} - 2x^T A^T Ax + 2y^T A^T Ax + 2(x - y)^T A^T b \\ &\geq \|Ax - b\|_2^2 - 2(x - y)^T \underbrace{A^T Ax}_{=A^T b} + 2(x - y)^T A^T b = \|Ax - b\|_2^2,\end{aligned}$$

also $\|Ax - b\|_2 \leq \|Ay - b\|_2$ und somit ist x auch eine Least-Squares-Lösung. \square

Übung 7.3 Zeigen Sie: Ist $B \in \mathbb{R}^{n \times n}$ eine symmetrische Matrix und $F(x) := x^T Bx$, $x \in \mathbb{R}^n$, dann ist $\nabla F(x) = 2Bx$. \diamond

Bemerkung 7.8 Die notwendige und hinreichende Bedingung (7.7) bedeutet, daß die Least-squares-Lösung so zu wählen ist, daß der **Residuenvektor** $r := Ax - b$ die Bedingung $A^T r = 0$ erfüllt, oder eben, daß $r \in \ker A^T$. Diese Bedingung hat eine geometrische Interpretation, siehe Abb. 7.2: Es sei

$$V := \mathcal{R}(A) := A\mathbb{R}^n := \{Ax : x \in \mathbb{R}^n\}$$

der von (den Spalten von) A erzeugte Vektorraum⁶². Dann ist das orthogonale Komplement

$$\mathbb{R}^n \ominus \mathcal{R}(A) = \left\{ x \in \mathbb{R}^n : x^T \mathcal{R}(A) = 0 \right\}$$

gerade $\ker A^T$. Somit sind die Least-Squares-Lösungen diejenigen Vektoren, bei denen der Residuenvektor senkrecht auf V steht.

Damit hätten wir bereits einen (naiven) Ansatz, um überbestimmte Gleichungssysteme zu lösen: wir verwenden das Cholesky-Verfahren aus Satz 5.4, um die Lösung von $A^T Ax = A^T b$ zu bestimmen, was zu einer Zerlegung

$$A^T A = GG^T$$

führt; dabei ist G aber ein *quadratischer* Faktor, im Gegensatz zu A . Allerdings führt der Übergang zu einem Gleichungssystem bezüglich $A^T A$, also die "Quadrierung" der Matrix auch zu einer Quadrierung des Fehlers.

⁶² $\mathcal{R}(A)$ steht für "Range" von A .

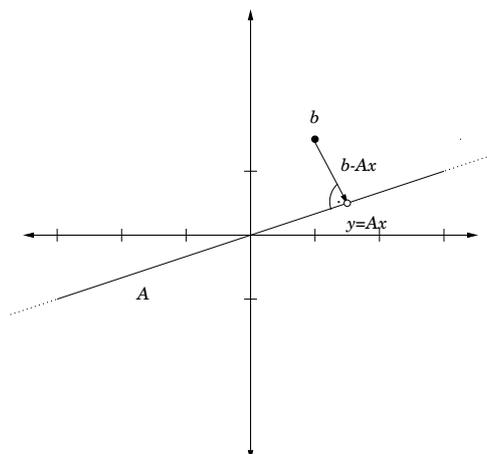


Abbildung 7.2: Geometrische Interpretation der Least-squares-Lösung: Der Fehler ist orthogonal zum Vektor Ax .

Beispiel 7.9 Es sei $A \in \mathbb{R}^{n+1 \times n}$ gegeben durch

$$A = \begin{bmatrix} 1 & 1 & \dots & 1 \\ \varepsilon & & & \\ & \varepsilon & & \\ & & \ddots & \\ & & & \varepsilon \end{bmatrix},$$

dann ist

$$A^T A = \begin{bmatrix} 1 & \varepsilon & & & \\ 1 & & \varepsilon & & \\ \vdots & & & \ddots & \\ 1 & & & & \varepsilon \end{bmatrix} \begin{bmatrix} 1 & 1 & \dots & 1 \\ \varepsilon & & & \\ & \varepsilon & & \\ & & \ddots & \\ & & & \varepsilon \end{bmatrix} = \begin{bmatrix} 1 + \varepsilon^2 & 1 & \dots & 1 \\ 1 & 1 + \varepsilon^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \dots & 1 & 1 + \varepsilon^2 \end{bmatrix}.$$

Diese Matrix ist quadratisch schlechter konditioniert als die Ausgangsmatrix! Denn da $B := A^T A = \mathbf{1}_n \mathbf{1}_n^T + \varepsilon^2 I$, ist jeder Vektor $y \in \mathbb{R}^n$ mit $\mathbf{1}_n^T y = 0$ ein Eigenvektor von B zum Eigenwert ε^2 . Außerdem ist

$$B \mathbf{1}_n = \mathbf{1}_n \underbrace{(\mathbf{1}_n^T \mathbf{1}_n)}_{=n} + \varepsilon^2 \mathbf{1}_n = (n + \varepsilon^2) \mathbf{1}_n.$$

Also ist, unter Verwendung des **Spektralradius**,

$$\kappa_2(A^T A) = \|A^T A\|_2 \left\| (A^T A)^{-1} \right\|_2 \geq \rho(A^T A) \rho\left((A^T A)^{-1}\right)$$

$$= \frac{\lambda_{\max}(A^T A)}{\lambda_{\min}(A^T A)} = 1 + \frac{n}{\varepsilon^2}.$$

Da jede Norm größer als der Spektralradius ist, ist die obige negative Aussage nicht etwa spezifisch für die 2–Norm, sondern gilt für jede beliebige Matrixnorm und damit auch für jede beliebige Konditionszahl.

In [17, S. 398, (19.14)] kann man diese quadratisch verschlechterte Situation auch konkreter finden: Für die berechnete Lösung \hat{x} des Least–Squares–Problems ergibt sich ein relativer Vorwärtsfehler von

$$\frac{\|x - \hat{x}\|_2}{\|x\|_2} \leq C_{m,n} \kappa_2^2(A) \hat{u}, \quad x \in \mathbb{R}^n \setminus \{0\}.$$

7.4 Singuläre–Werte–Zerlegung und die Pseudoinverse

Wir wollen uns nun der Beschreibung von Least–squares–Lösungen etwas theoretischer nähern.

Definition 7.10 Eine quadratische Matrix $U \in \mathbb{R}^{n \times n}$ bzw. $U \in \mathbb{C}^{n \times n}$ heißt *orthogonal* bzw. *unitär*, wenn $U^T U = I$ bzw. $U^H U = I$. Eine Matrix $D \in \mathbb{R}^{m \times n}$ heißt *diagonal*, wenn $a_{jk} = 0$ falls $j \neq k$, $j = 1, \dots, m$, $k = 1, \dots, n$.

Bemerkung 7.11 Die orthogonalen Matrizen bilden eine multiplikative Gruppe: Jede orthogonale Matrix U hat eine orthogonale Inverse (nämlich U^T) und das Produkt zweier orthogonaler Matrizen U, V ist wieder orthogonal (da $(UV)^T UV = V^T U^T UV = V^T V = I$).

Satz 7.12 Zu jeder Matrix $A \in \mathbb{R}^{m \times n}$, $m, n \in \mathbb{N}$, gibt es orthogonale Matrizen $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$ und eine Diagonalmatrix $\Sigma \in \mathbb{R}^{m \times n}$ so daß

$$A = U \Sigma V^T, \quad \sigma_{11} \geq \sigma_{22} \geq \dots \geq \sigma_{\ell, \ell} \geq 0, \quad \ell := \min(m, n). \quad (7.8)$$

Definition 7.13 Die Darstellung (7.8) heißt *Singuläre–Werte–Zerlegung* bzw. *Singular Value Decomposition (SVD)*, die Diagonalwerte von Σ , $\sigma_j = \sigma_{jj}$, $j = 1, \dots, \ell$, heißen *Singulärwerte* von A .

Beweis: Induktion über ℓ . Im Fall $\ell = 1$ nehmen wir ohne Einschränkung⁶³ an, daß $m \leq n$. Dann hat A die Form $A = [a_{11} \dots a_{1n}] =: a^T$. Ist $a = 0$, dann bilden $U = 1$, $\Sigma = 0$ und $V = I$ die SVD von A . Andernfalls setzen wir $v_1 = \|a\|_2^{-1} a$, also als normierte Version von a und vervollständigen diesen Vektor zu einer Orthonormalbasis des \mathbb{R}^n , das heißt, wir wählen v_2, \dots, v_n so, daß

$$\mathbb{R}^n = \text{span} \{v_1, \dots, v_n\}, \quad v_j^T v_k = \delta_{jk}, \quad j, k = 1, \dots, n,$$

⁶³Ansonsten gehen wir zu A^T über.

und erhalten die Komponenten der SVD als

$$U = 1, \quad V = [v_j : j = 1, \dots, n], \quad \Sigma = U^T A V = [\|a\|_2 \ 0 \ \dots \ 0].$$

Sei also Satz 7.12 für ein $\ell > 0$ bewiesen und seien $m, n \in \mathbb{N}$ dergestalt, daß $\min(m, n) = \ell + 1$. Wir setzen

$$\tau_1 := \max_{x \in \mathbb{R}^n} \frac{\|Ax\|_2^2}{\|x\|_2^2} = \max_{\|x\|_2=1} x^T (A^T A) x. \quad (7.9)$$

Ist $\tau_1 = 0$, dann ist $A = 0$ und die SVD ergibt sich als⁶⁴ $U = I, V = I, \Sigma = 0$. Andernfalls ist τ_1 der größte Eigenwert der symmetrischen positiv definiten Matrix $A^T A$; sei außerdem $v_1 \in \mathbb{R}^n$ ein zugehöriger normierter Eigenvektor. Diesen kann man wieder mit v_2, \dots, v_n zu einer Orthonormalbasis des \mathbb{R}^n ergänzen und damit ist die Matrix $V = [v_j : j = 1, \dots, n]$ orthogonal. Wegen (7.9) ist $\|Av_1\|_2 = \sigma_1, \sigma_1 = \sqrt{\tau_1}$, und damit ist

$$0 \neq u_1 := \frac{1}{\sigma_1} A v_1 \in \mathbb{R}^m, \quad \|u_1\|_2 = 1.$$

Wie vorher vervollständigen wir diesen Vektor zu einer Orthonormalbasis, diesmal mit u_2, \dots, u_m und setzen $U = [u_j : j = 1, \dots, m]$. Damit ist

$$U^T A V = \begin{bmatrix} \sigma_1 & 0 \\ 0 & B \end{bmatrix} \iff A = U \begin{bmatrix} \sigma_1 & 0 \\ 0 & B \end{bmatrix} V^T, \quad B \in \mathbb{R}^{m-1 \times n-1}, \quad (7.10)$$

wobei der größte Eigenwert von $B^T B$ höchstens σ_1 ist. Nach Induktionsannahme gibt es Matrizen $\hat{U} \in \mathbb{R}^{m-1 \times m-1}, \hat{V} \in \mathbb{R}^{n-1 \times n-1}$ und $\hat{\Sigma} \in \mathbb{R}^{m-1 \times n-1}$, so daß

$$B = \hat{U} \hat{\Sigma} \hat{V}^T, \quad \sigma_1 \geq \hat{\sigma}_1 \geq \dots \geq \hat{\sigma}_{\ell-1} \geq 0.$$

Setzt man das in (7.10) ein, so ergibt sich, daß

$$A = \underbrace{U}_{=:U} \begin{bmatrix} 1 & 0 \\ 0 & \hat{U} \end{bmatrix} \underbrace{\begin{bmatrix} \sigma_1 & 0 \\ 0 & \hat{\Sigma} \end{bmatrix}}_{=: \Sigma} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & \hat{V}^T \end{bmatrix}}_{=: V^T} V^T,$$

womit der Beweis fertig ist. □

Bemerkung 7.14 (Beobachtungen zur SVD)

1. Der singuläre Wert τ_1 aus (7.9) ist gleichzeitig der größte Eigenwert der symmetrischen positiv semidefiniten Matrix $A^T A$: Seien $\lambda_1 \geq \dots \geq \lambda_n \geq 0$ die Eigenwerte von $B = A^T A$ und x_1, \dots, x_n zugehörige orthonormale Eigenvektoren, dann ist

$$x^T B x = \left(\sum_{j=1}^n \alpha_j x_j \right)^T B \left(\sum_{j=1}^n \alpha_j x_j \right) = \sum_{j=1}^n \alpha_j^2 \lambda_j \leq \lambda_1 \sum_{j=1}^n \alpha_j^2 = \lambda_1 \|x\|_2^2.$$

⁶⁴Achtung: U und V können Einheitsmatrizen unterschiedlicher Größe sein.

2. Generell sind die singulären Werte einer Matrix A die (immer noch nichtnegativen) Wurzeln der Eigenwerte von $A^T A$.
3. Die Idee der SVD wird aus dem Beweis ersichtlich: man bestimmt orthogonale Vektoren $u_1, \dots, u_m \in \mathbb{R}^m, v_1, \dots, v_n \in \mathbb{R}^n$, so daß die von ihnen gebildeten Matrizen U und V die Ausgangsmatrix A dergestalt orthogonal transformieren, daß

$$\left(U^T A V \right) e_j = \sigma_j e_j, \quad j = 1, \dots, \ell,$$

also die kanonischen Einheitsvektoren in sich selbst abbilden.

4. Die SVD ist nicht immer eindeutig! Das einfachste Gegenbeispiel ist $A = 0$, dann ist $\Sigma = 0$ und man kann U und V sogar frei wählen. Was natürlich eindeutig ist⁶⁵ ist die Matrix Σ .

Definition 7.15 Die **Moore–Penrose–Inverse** oder **Pseudoinverse** einer $m \times n$ -Matrix A mit SVD $A = U \Sigma V^T$ ist definiert als

$$A^+ = V \Sigma^+ U^T, \quad \Sigma^+ \in \mathbb{R}^{n \times m}$$

wobei

$$(\Sigma^+)_{jk} = \begin{cases} \sigma_{jk}^{-1}, & \sigma_{jk} \neq 0, \\ 0 & \sigma_{jk} = 0, \end{cases} \quad j = 1, \dots, n, k = 1, \dots, m.$$

Der Name ‘‘Pseudoinverse’’ ist leicht erklärt. Ist nämlich $A \in \mathbb{R}^{n \times n}$ eine invertierbare Matrix, dann sind wegen

$$0 < |\det A| = \underbrace{|\det U|}_{=1} |\det \Sigma| \underbrace{|\det V|}_{=1} = \prod_{j=1}^n \sigma_j$$

alle singulären Werte von A strikt positiv und damit ist $\Sigma^+ = \Sigma^{-1}$. Daher ist

$$A A^+ = U \Sigma V^T V \Sigma^{-1} U^T = I = V \Sigma^{-1} U^T U \Sigma V^T = A^+ A$$

und damit ist $A^+ = A^{-1}$. Im Falle einer invertierbaren Matrix kann man das ‘‘Pseudo–’’ also getrost weglassen.

Satz 7.16 Sei $A \in \mathbb{R}^{m \times n}, m > n$.

1. Die Pseudoinverse A^+ erfüllt die **Moore–Penrose–Bedingungen**

$$(a) A^+ A = (A^+ A)^T,$$

$$(b) A A^+ = (A A^+)^T,$$

⁶⁵Wegen der absteigenden Anordnung der singulären Werte.

Bemerkung 7.17 *In gewissem Sinne ist die Pseudoinverse ein Lieblingskind der Ingenieure, die schon vor langer Zeit herausgefunden haben, daß der Befehle `pinv` in `Matlab/Octave` immer eine "Inverse" liefert, aber nie eine dieser nervigen Fehlermeldungen oder Warnungen. Daß dafür die Ergebnisse nicht sonderlich sinnvoll sein können⁶⁶ nimmt man oftmals gerne in Kauf. Es braucht durchaus einiges an Überzeugungsarbeit, um dafür zu sorgen, daß die Pseudoinverse nicht in unangebrachter Weise verwendet⁶⁷ wird!*

7.5 Orthogonaltransformationen

In diesem Kapitel wollen wir Transformationsmethoden (eine Art Elimination) unter Verwendung *orthogonaler* Matrizen betrachten, um so die Lösung des Ausgleichsproblems *direkt* aus der Matrix A zu ermitteln. Warum man gerade orthogonale Matrizen verwendet, ist eigentlich ziemlich naheliegend: weil für jede orthogonale Matrix $U \in \mathbb{R}^{n \times n}$

$$\|Ux\|_2 = \sqrt{(Ux)^T(Ux)} = \sqrt{x^T U^T U x} = \|x\|_2, \quad x \in \mathbb{R}^n,$$

gilt, die 2-Norm also **orthogonal invariant** ist, ist auch die Matrixnorm $\|\cdot\|$ orthogonal invariant⁶⁸ und damit ist auch

$$\kappa_2(UA) = \kappa_2(AV) = \kappa_2(A), \quad A \in \mathbb{R}^{m \times n}, U \in \mathbb{R}^{m \times m}, V \in \mathbb{R}^{n \times n},$$

Multiplikation mit orthogonalen Matrizen verschlechtert also nicht die Kondition des Problems⁶⁹. Was wir also brauchen, sind einfache orthogonale Matrizen, mit denen man „etwas anfangen kann“.

Definition 7.18 *Zu $0 \neq y \in \mathbb{R}^m$ ist die **Householder-Matrix** $H = H(y)$ definiert als*

$$H(y) = I - 2 \frac{yy^T}{\|y\|_2^2}.$$

Die Idee, Householder-Matrizen zur Lösung von Least-squares-Problemen zu verwenden geht⁷⁰ auf Householder [18] zurück, die praktischen Details wurden in [14] ausgearbeitet.

Lemma 7.19 *Householder-Matrizen sind symmetrisch und orthogonal: $H^T = H$ und $H^T H = I$.*

⁶⁶Bei der Bildung der Pseudoinversen muss entschieden werden, ab wann ein singulärer Wert als $= 0$ angesehen werden soll – da steckt dann auch das gesamte numerische Teufelswerk!

⁶⁷Genauso unangebracht wie das grammatistisch inkorrekte "verwandt" an dieser Stelle.

⁶⁸Entsprechendes gilt auch immer für komplexe Matrizen und *unitäre* Invarianz

⁶⁹Und den Traum, daß sich während der Rechnung ein Problem in ein besser konditioniertes umwandelt, sollte man schon in der Numerik I aufgegeben haben.

⁷⁰Welch ein Wunder ...

Beweis: Symmetrie ist trivial und die Orthogonalität folgt aus

$$H^T H = H^2 = \left(I - \frac{2}{\|y\|_2^2} y y^T \right)^2 = I - \frac{4}{\|y\|_2^2} y y^T + \frac{4}{\|y\|_2^2} y \underbrace{\frac{y^T y}{\|y\|_2^2}}_{=1} y^T = I.$$

□

Der Einfachheit halber wollen wir nun immer annehmen, daß der **Householder-Vektor** y für die Bildung der Householder-Matrix $H(y)$ **normiert** ist, das heißt $\|y\|_2 = 1$, denn dann ist $H(y) = I - 2yy^T$.

Bemerkung 7.20 Die Householder-Transformation $x \mapsto H(y)x$ kann man auch geometrisch interpretieren, und zwar als **Spiegelung** an der Hyperebene

$$E := \left\{ x \in \mathbb{R}^m : x^T y = 0 \right\},$$

siehe Abb. 7.3. Schreibt man nämlich x als $x = \alpha y + y^\perp$, $y^T y^\perp = 0$, dann ist⁷¹

$$H(y) x = \left(I - 2yy^T \right) \left(\alpha y + y^\perp \right) = \alpha y + y^\perp - 2\alpha y \underbrace{y^T y}_{=1} - 2y \underbrace{y^T y^\perp}_{=0} = -\alpha y + y^\perp,$$

das heißt, die Projektion von x in die Hyperebene E bleibt unverändert, während das Vorzeichen des Anteils in Richtung y umgedreht wird.

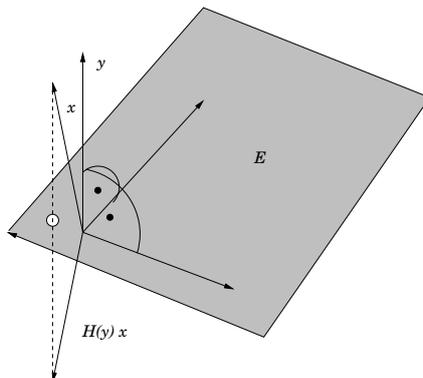


Abbildung 7.3: Die Householder-Transformation als Spiegelung an der Hyperebene $E = y^\perp$. Im physikalischen Sinn ist das also eher eine Brechung als eine Spiegelung: "Einfallswinkel gleich Ausfallswinkel".

⁷¹Nicht vergessen: $\|y\|_2 = 1$!

Sei also $A \in \mathbb{R}^{m \times n}$, $m > n$. Die "Strategie" ist nun naheliegend: Wir wollen mit Hilfe von Householder-Matrizen wie bei der Gauß-Elimination A auf eine obere Dreiecksgestalt bringen, d.h. *normierte* Vektoren y_1, \dots, y_k finden, so daß

$$H(y_k) \cdots H(y_1) A = \left[\begin{array}{ccc|ccc} * & \dots & * & * & * & \dots & * \\ & & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & & * & * & * & \dots & * \\ \hline & & & \mathbf{a}_{k+1}^{(k+1)} & * & \dots & * \\ & & & \vdots & \vdots & \ddots & \vdots \\ & & & \mathbf{a}_m^{(k+1)} & * & \dots & * \end{array} \right], \quad k = 0, \dots, n-1. \quad (7.12)$$

Für $k = 0$ ist (7.12) trivialerweise erfüllt. Um von (7.12) einen Schritt weiterzukommen, müssen wir also einen Vektor $\hat{y}_{k+1} \in \mathbb{R}^{m-k}$ und eine Zahl $\alpha_{k+1} \in \mathbb{R}$ bestimmen, so daß⁷²

$$H(\hat{y}_{k+1}) \mathbf{a}^{(k+1)} = \alpha_{k+1} \mathbf{e}_1 \in \mathbb{R}^{m-k}, \quad \mathbf{a}^{(k+1)} = \begin{bmatrix} \mathbf{a}_{k+1}^{(k+1)} \\ \vdots \\ \mathbf{a}_m^{(k+1)} \end{bmatrix} \in \mathbb{R}^{m-k}, \quad (7.13)$$

denn ergibt sich unter Verwendung von

$$\mathbf{y}_{k+1} := \begin{bmatrix} 0 \\ \hat{y}_{k+1} \end{bmatrix} \in \mathbb{R}^m \quad \Rightarrow \quad H(\mathbf{y}_{k+1}) = \begin{bmatrix} I_k & 0 \\ 0 & H(\hat{y}_{k+1}) \end{bmatrix} \in \mathbb{R}^{m \times m},$$

daß

$$H(\mathbf{y}_{k+1}) \cdots H(\mathbf{y}_1) A = \left[\begin{array}{ccc|ccc} * & \dots & * & * & * & \dots & * \\ & & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & & * & * & * & \dots & * \\ \hline & & & \alpha_{k+1} & * & \dots & * \\ & & & 0 & * & \dots & * \\ & & & \vdots & \vdots & \ddots & \vdots \\ & & & 0 & * & \dots & * \end{array} \right].$$

Das entscheidende Problem besteht also darin, (7.13) zu lösen. Daß und wie das geht, sagt uns das nächste Lemma.

Lemma 7.21 Sei $x \in \mathbb{R}^m$. Setzt man⁷³ $\gamma = 2 \|x\|_2$, $\epsilon = -\operatorname{sgn} x_1$ und

$$\mathbf{y}_1 = \sqrt{\frac{1}{2} + \frac{|x_1|}{\gamma}}, \quad \mathbf{y}_j = -\epsilon \frac{x_j}{\gamma \mathbf{y}_1}, \quad j = 2, \dots, m, \quad (7.14)$$

dann ist $\|\mathbf{y}\|_2 = 1$ und $H(\mathbf{y}) x = \alpha \mathbf{e}_1$, wobei $\alpha = -\epsilon \|x\|_2$.

```
##  
## HousehStep.m  
## Berechne y-Vektor fuer Householder-Matrix  
## Daten:  
## x Zu transformierender Vektor  
  
function y = HousehStep( x )  
    m = length( x );  
    y = zeros( m,1 );  
  
    g = 2 * sqrt( x' * x );  
  
    if g < eps  
        return;  
    endif  
  
    s = sign( x(1) );  
    if s = 0                % Sonderfall !!!  
        s = 1;  
    end  
  
    y( 1 ) = sqrt( 0.5 + abs( x(1) )/g );  
    gg = g * y( 1 );  
    for j = 2:m  
        y( j ) = s * x( j ) / gg;  
    end  
endfunction
```

Programm 7.1 HousehStep.m: Bestimmung des Vektors y für die Householder-Matrix. Die Behandlung des Sonderfalls $x = 0$ wird noch gerechtfertigt werden.

```
##
## HousehStep.m
## Berechne y-Vektor fuer Householder-Matrix
## Daten:
## x Zu transformierender Vektor

function y = HousehStep2( x )
    m = length( x );
    y = zeros( m,1 );

    g = 2 * sqrt( x' * x );

    if g < eps
        return;
    endif

    if x(1) == 0          % Sonderfall !!!
        s = 1;
    else
        s = sign( x(1) );
    end

    y( 1 ) = sqrt( 0.5 - abs(x(1))/g );
    gg = g * y( 1 );
    for j = 2:m
        y( j ) = -s * x( j ) / gg;
    end
endfunction
```

Programm 7.2 HousehStep2.m: Die "dumme" Art, den Vektor y zu bestimmen, mit der numerisch falschen Wahl von ϵ .

Übung 7.5 Folgern Sie aus (7.14), daß $\|y\|_2 = 1$. ◇

Beweis: Natürlich könnte man "einfach" nachrechnen, daß die Bedingungen aus (7.14) die gewünschten Eigenschaften liefern. Nachdem aber diese Voraussetzungen ja normalerweise nicht vom Himmel fallen, wollen wir sie explizit herleiten – das ist zwar etwas aufwendiger, aber auch "realistischer" und zeigt uns, wo man numerisch aufpassen muss.

Da $H(y)$ orthogonal ist, muß zuerst einmal

$$\|\alpha\| = \|H(y) x\|_2 = \|x\|_2$$

gelten, das heißt $\alpha = \epsilon \|x\|_2$, $\epsilon \in \{\pm 1\}$. Nach der Definition von $H(y)$ ist $H(y) x = \alpha e_1$ äquivalent zu

$$x_1 - 2y_1 (y^T x) = \alpha, \quad x_j - 2y_j (y^T x) = 0, \quad j = 2, \dots, m. \quad (7.15)$$

Aus diesen Bedingungen ergibt sich mit $\beta := y^T x$ und der Zusatzforderung $\|y\|_2 = 1$, daß

$$\begin{aligned} \alpha y_1 &= y_1 \underbrace{(x_1 - 2y_1 \beta)}_{=\alpha} + \sum_{j=2}^m y_j \underbrace{(x_j - 2y_j \beta)}_{=0} = \sum_{j=1}^m x_j y_j - 2\beta \sum_{j=1}^m y_j^2 \\ &= \beta - 2\beta \underbrace{\|y\|_2^2}_{=1} = -\beta, \end{aligned}$$

also $\beta = -\alpha y_1$. Eingesetzt in die erste Gleichung von (7.15) liefert dies, daß

$$x_1 + 2\alpha y_1^2 = \alpha \quad \implies \quad y_1 = \sqrt{\frac{1 - x_1/\alpha}{2}}.$$

Wählen wir nun $\epsilon \in \{-1, 1\}$ so, daß

$$\epsilon x_1 < 0, \quad \implies \quad x_1 = -\epsilon |x_1| \quad (7.16)$$

dann ergibt sich

$$y_1 = \sqrt{\frac{1 - \frac{-\epsilon |x_1|}{\epsilon \|x_2\|}}{2}} = \sqrt{\frac{1 + \frac{|x_1|}{\|x\|_2}}{2}} \quad \implies \quad \beta = -\epsilon \|x\|_2 y_1$$

und schließlich

$$y_j = \frac{x_j}{2\beta} = -\epsilon \frac{x_j}{2y_1 \|x\|_2}, \quad j = 2, \dots, m.$$

□

⁷²Hierbei bezeichnet e_j wieder den j -ten kanonischen Einheitsvektor.

⁷³Hierbei können wir das Vorzeichen $\text{sgn } 0$ der Zahl Null beliebig als ± 1 wählen.

```

%#
%# HousehMat.m
%# Berechne Householder-Matrix  $H(y) = I - 2 y y^T / y^T y$ 
%# Daten:
%#   y   Vektor

function H = HousehMat( y )
    H = eye( length(y) ) - 2*y*y'/(y'*y);

```

Programm 7.3 HousehMat.m: Berechnung der Householder-Matrix zu y .

Bemerkung 7.22 Die Wahl von ϵ in (7.16) war theoretisch beliebig, aber numerisch absolut sinnvoll, denn nun werden unter der Wurzel bei der Bestimmung von y_1 zwei positive Zahlen addiert, so daß **Auslöschung** vermieden wird, siehe Beispiel 2.9. Und da in (7.14) durch y_1 dividiert wird, sollten wir ohnehin dafür sorgen, daß diese Zahl so groß wie möglich wird, einfach um die Fehlerfortpflanzung möglichst klein zu halten.

Die maximale Auslöschung bei "falschen" ϵ träte übrigens genau dann auf, wenn $|x_1| \sim \|x_2\|$ wäre, also genau dann, wenn der Vektor x sowieso schon den Wert hätte, den er haben sollte. Und das kann man tatsächlich auch deutlich sehen wie in Beispiel 7.23 gezeigt werden wird. an kann die Bestimmung des Householder-Vektors aber auch wie in Abb. 7.4 geometrisch interpretieren. Im \mathbb{R}^2 sind die Spiegelungsebenen die beiden Winkelhalbierenden der von x und der x_1 -Achse erzeugten Winkel; diese Winkel summieren sich natürlich zu 2π . Wird nun einer der Winkel immer kleiner, dann ist zwar die Spiegelung am spitzen Winkel immer unproblematischer, dafür wird aber die Spiegelung an der anderen Geraden eine immer kritischere Operation, und am schlimmsten ist das ganze, wenn einer der Winkel gegen 0 geht. Keinen Unterschied gibt es, wenn $x_1 = 0$ ist, also in dem Fall, in dem wir ϵ nach Gusto wählen können, weil wir $\text{sgn } 0$ beliebig in ± 1 festlegen dürfen.

Beispiel 7.23 Vergleichen wir einmal für $n = 2$ die beiden Routinen aus den Programmen 7.1 und 7.2, und zwar zuerst für eine eher braven Vektor:

```

octave> x = [ 1 1 ]';
octave> y1 = HousehStep (x); y2 = HousehStep2 (x);
octave> HousehMat (y1)*x
ans =

    -1.4142e+00
    -1.1102e-16

octave> HousehMat (y2)*x
ans =

```

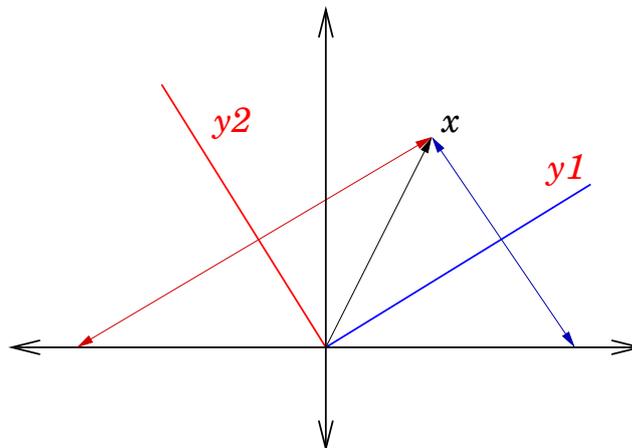


Abbildung 7.4: Die geometrische Interpretation der **Householder-Transformation**: x wird an den beiden Winkelhalbierenden gespiegelt, und das "gute" ϵ entspricht dem spitzen, das schlechte dem stumpfen Winkel.

```
1.4142e+00
4.4409e-16
```

und in beiden Fällen wird x praktisch auf ein Vielfaches des Einheitsvektors abgebildet⁷⁴. Das wird aber anders, wenn wir den Fall

```
octave> x = [ 1 10^(-5) ]';
octave> y1 = HousehStep (x); y2 = HousehStep2 (x);
```

betrachten, denn nun erhalten wir das schon signifikant schlechtere Ergebnis

```
octave> HousehMat( y1 ) * x
ans =
```

```
-1.0000e+00
1.1911e-22
```

```
octave> HousehMat( y2 ) * x
ans =
```

```
1.0000e+00
8.2815e-13
```

Noch interessanter wird es für

⁷⁴Daß im zweiten Fall der Fehler ein wenig größer ist hat noch nicht wirklich etwas zu sagen.

```
octave> x = [ 1 .2*10^(-7) ]'; y1 = HousehStep (x); y2 = HousehStep2 (x);
octave> HousehMat( y1 ) * x
ans =
```

```
-1.0000e+00
-1.1325e-24
```

```
octave> HousehMat( y2 ) * x
ans =
```

```
1.0000e+00
2.2045e-09
```

und richtig übel im Fall

```
octave> x = [ 1 .18*10^(-7) ]'; y2 = HousehStep2 (x);
warning: in /.../HousehStep2.m near line 26, column 12:
```

```
>>> y (j) = -s * x (j) / gg;
```

```
warning: division by zero
```

```
octave> HousehMat( y2 ) * x
ans =
```

```
NaN
NaN
```

wo die Auslöschung absolut gnadenlos zuschlägt.

Unabhängig von den numerischen Fußangeln haben wir auch schon wieder eine neue Zerlegung für Matrizen kennengelernt, die sogenannte *QR-Zerlegung*.

Satz 7.24 Sei $A \in \mathbb{R}^{m \times n}$, $m \geq n$. Dann gibt es eine *orthogonale Matrix* $Q \in \mathbb{R}^{m \times m}$ und eine *Rechtsdreiecksmatrix*⁷⁵

$$R = \begin{bmatrix} * & \dots & * \\ & \ddots & \vdots \\ 0 & & * \\ \hline 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} \in \mathbb{R}^{m \times n},$$

⁷⁵Nachdem U für unitäre Matrizen reserviert sein wird, ist R jetzt der Ersatz für unser gutes altes U aus der LU-Zerlegung von oben. So ein kleiner Notationswechsel zwischendurch erhält die geistige Flexibilität.

```

%#
%# HousehQR.m
%# Berechne QR-Zerlegung fuer Matrix A
%# Daten:
%# A Zu zerlegende Matrix, ueberschreibe mit R

function [ Q,R ] = HousehQR( A )
    [m,n] = size( A );
    Q = II = eye( m );

    for j = 1:n
        x = A( j:m,j );
        y = HousehStep( x );           % Berechne orthog. Vektor
        if j > 1
            y = [ zeros( j-1,1 ); y ]; % Auffuellen
        end
        H = II - 2*y*y';              % Householder-Matrix

        A = H * A;
        Q = Q * H;
    end

    R = A;
%endfunction

```

Programm 7.4 HousehQR.m: QR-Zerlegung mit Householder-Matrizen.

so daß $A = QR$.

Beweis: Wie man die Vektoren y_1, \dots, y_{n-1} bestimmt, so daß

$$H(y_{n-1}) \cdots H(y_1) A = R$$

wissen wir ja bereits aus Lemma 7.21 – solange der “eingeebene” Vektor x nicht der Nullvektor ist. Finden wir aber bei unserer Elimination eine “Nullspalte” vor, dann ist das auch nicht schlimm, denn die obere Dreiecksstruktur ist ja dann insbesondere vorhanden und wir können in diesem Fall einfach $y = 0$ setzen. Bleibt nur noch zu bemerken, daß

$$Q = H(y_1)^T \cdots H(y_{n-1})^T = H(y_1) \cdots H(y_{n-1}).$$

□

Was bringt uns nun die QR-Zerlegung? Nun, mit $A = QR$ wird das Normalgleichungssystem zu

$$R^T Q^T b = A^T b = A^T A x = R^T Q^T Q R x = R^T R x.$$

Schreiben wir

$$R = \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix}, \quad \hat{R} \in \mathbb{R}^{n \times n},$$

dann ist

$$\begin{bmatrix} \hat{R}^T & 0 \end{bmatrix} Q^T b = \begin{bmatrix} \hat{R}^T & 0 \end{bmatrix} \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix} x = \hat{R}^T \hat{R} x.$$

Hat nun A (und damit auch R und \hat{R}) vollen Rang, dann können wir von links mit \hat{R}^{-T} multiplizieren und erhalten das einfache Gleichungssystem

$$\hat{R}x = \hat{R}^{-T} \begin{bmatrix} \hat{R}^T & 0 \end{bmatrix} Q^T b = [I_n \ 0] Q^T b = \begin{bmatrix} (Q^T b)_1 \\ \vdots \\ (Q^T b)_n \end{bmatrix},$$

das wir direkt durch **Rücksubstitution** lösen können. Das fassen wir noch einmal zusammen.

Korollar 7.25 Die Matrix $A \in \mathbb{R}^{m \times n}$, $m \geq n$, habe vollen Rang und die QR-Zerlegung $A = QR$, $R = \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix}$. Dann erhält man die **Least-Squares-Lösung** von (7.6) als Lösung des Gleichungssystems⁷⁶

$$\hat{R}x = KQ^T b, \quad K = \sum_{j=1}^n e_j e_j^T \in \mathbb{R}^{n \times m}. \quad (7.17)$$

Hat die Matrix A nicht vollen Rang, dann hat man aber, auch mit der QR-Zerlegung, ein Problem! Denn um die Gleichung

$$\hat{R}^T Q^T b = \hat{R}^T \hat{R} x$$

durch Vorwärtselemination und Rücksubstitution lösen zu können, müssen ja \hat{R} und damit auch \hat{R}^T vollen Rang haben, insbesondere müssen die Diagonalelemente von \hat{R} von Null verschieden sein. Ein Diagonalwert 0 in der Matrix R tritt aber nach Lemma 7.21 genau dann auf, wenn während der Householder-Elimination eine **Nullspalte** unter der Diagonalen erzeugt wurde, siehe auch die Bemerkung im Beweis von Satz 7.24. Was man mit Nullspalten macht, wissen wir aber aus der Gauß-Elimination, Stichwort **Pivotsuche**: Nach Anwendung einer Spaltenpermutation können wir entweder mit

⁷⁶Die Matrix K hier ist die **kanonische Koordinatenprojektion** vom \mathbb{R}^m in den \mathbb{R}^n .

```

%#
%# HousehLS.m
%# Loese LS-Problem
%#   Ax - b = min
%# ueber QR-Zerlegung mit Householder-Matrizen
%# Daten:
%#   A Matrix, voller Rang!
%#   b rechte Seite

function x = HousehLS( A,b )
    [ m,n ] = size( A );
    [ Q,R ] = HousehQR( A );

    R = R( 1:n, 1:n );      % LS-relevanter Teil
    b = Q' * b;

    x = RueckSubs( R , b( 1:n ) );
%endfunction

```

Programm 7.5 HousehLS.m: Lösung des Least-squares-Problems nach (7.17).

einer von Null verschiedenen Spalte weitermachen, oder aber die eliminierte Matrix hat die Form

$$\begin{bmatrix} R & B \\ 0 & 0 \end{bmatrix}, \quad R \in \mathbb{R}^{r \times r}, B \in \mathbb{R}^{r \times n-r}, \quad r = \text{rang } A.$$

Da für Spaltenvertauschungen eine Permutationsmatrix von rechts multipliziert werden muß, hat also die QR-Zerlegung mit Spaltenpivotsuche die Form

$$AP = QR, \quad P \in \Pi_n,$$

wobei wieder $\Pi_n \subset \mathbb{R}^{n \times n}$ die Gruppe der Permutationsmatrizen bezeichnet. Die Details zur QR-Zerlegung mit Pivotsuche finden sich beispielsweise in [13, Sec. 6.4].

Zum Schluß noch ein Wort zur **Stabilität** der QR-Zerlegung, die so ziemlich genauso ist, wie man es von der Gauß-Elimination erwarten würde. Unter Verwendung der **Frobenius-Norm**

$$\|A\|_F = \left(\sum_{j=1}^m \sum_{k=1}^n a_{jk}^2 \right)^{1/2}, \quad A \in \mathbb{R}^{m \times n}$$

findet man in [17, Theorem 18.4] das folgende Resultat.

Satz 7.26 Es sei $A \in \mathbb{R}^{m \times n}$, $m \geq n$, und $\hat{R} \in \mathbb{R}^{m \times n}$ die mit dem Householder–Verfahren berechnete Rechtsdreiecksmatrix⁷⁷ der QR–Zerlegung von A . Dann gibt es eine orthogonale Matrix $\hat{Q} \in \mathbb{R}^{n \times n}$ und eine Matrix $\hat{A} \in \mathbb{R}^{m \times n}$, so daß

$$\hat{A} = \hat{Q}\hat{R}$$

und

$$\|A - \hat{A}\|_F \leq C nm\hat{u} \|A\|_F + O(nm\hat{u}^2)$$

wobei $C > 0$ eine moderate Konstante ist.

Den Beweis schenken wir uns ...

7.6 Orthogonaltransformationen II – Givens–Rotationen

Eine Alternative sind die sogenannten **Givens–Rotationen** [11], wobei die Idee der Rotationsmatrizen sogar schon auf Jacobi [20] zurückgeht⁷⁸. Um diesen Zugang zu motivieren, sehen wir uns einmal an, wie orthogonale Matrizen eigentlich im 2×2 –Fall aussehen. Sei also

$$Q = \begin{bmatrix} q & r \\ s & t \end{bmatrix}$$

eine orthogonale Matrix, das heißt,

$$I = Q^T Q = \begin{bmatrix} q & s \\ r & t \end{bmatrix} \begin{bmatrix} q & r \\ s & t \end{bmatrix} = \begin{bmatrix} q^2 + s^2 & qr + st \\ qr + st & r^2 + t^2 \end{bmatrix} \quad (7.18)$$

Damit folgen aus (7.18) die Beziehungen

$$1 = q^2 + s^2 = r^2 + t^2, \quad 0 = qr + st.$$

Die erste Identität sagt uns daß wir q, r, s, t parametrisieren können⁷⁹, und zwar *bis auf Vorzeichen* als

$$q = \cos \phi, \quad s = \sin \phi, \quad r = -\sin \psi, \quad t = \cos \psi, \quad \phi, \psi \in [0, \pi).$$

Diese Vorzeichen, die genaugenommen beispielsweise $q = \pm \cos \phi$ liefern würden, repräsentieren den *Spiegelungsanteil*. Aus der zweiten Beziehung folgt unter Verwendung der Additionstheoreme, daß

$$0 = \cos \phi \sin \psi - \sin \phi \cos \psi = \sin(\phi - \psi),$$

⁷⁷Achtung: hier hat \hat{R} eine andere Bedeutung als weiter oben in diesem Abschnitt

⁷⁸Woran man wieder einmal sieht, daß die „Alten“ sich auch für Numerik interessiert haben.

⁷⁹Die Einschränkung von ϕ und ψ kommt daher daß Q trivialerweise genau dann orthogonal ist, wenn $-Q$ orthogonal ist.

Drehwinkel ϕ_{12} , so daß

$$J(1,2; \phi_{12}) A = \begin{bmatrix} * & * & \dots & * \\ 0 & * & \dots & * \\ * & * & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \dots & * \end{bmatrix},$$

dann einen Drehwinkel ϕ_{13} , so daß

$$J(1,3; \phi_{13}) J(1,2; \phi_{12}) A = \begin{bmatrix} * & * & \dots & * \\ 0 & * & \dots & * \\ 0 & * & \dots & * \\ * & * & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \dots & * \end{bmatrix},$$

und so weiter, bis schließlich⁸¹

$$\left(\prod_{s=m}^2 J(1,s; \phi_{1s}) \right) A = \begin{bmatrix} * & * & \dots & * \\ 0 & * & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & * & \dots & * \end{bmatrix}$$

Als nächstes eliminieren wir in der zweiten Spalte mit Hilfe von Drehwinkeln $\phi_{23}, \dots, \phi_{2m}$ die Subdiagonalelemente. Da außerdem, für jedes $B \in \mathbb{R}^{m \times n}$ mit $b_{21} = \dots = b_{m1} = 0$ und $r = 2, \dots, m$,

$$(J(2,k; \phi_{2k})B)_{r1} = \begin{cases} b_{r1} & r \neq 2, k \\ \cos \phi_{2k} b_{21} + \sin \phi_{2k} b_{k1} & r = 2 \\ -\sin \phi_{2k} b_{21} + \cos \phi_{2k} b_{k1} & r = k \end{cases} = 0, \quad k = 3, \dots, m,$$

die Nullen in der ersten Spalte bleiben also unangetastet. Damit ist die Vorgehensweise klar: Man bestimmt Drehwinkel

$$\phi_{12}, \dots, \phi_{1m}, \phi_{23}, \dots, \phi_{2m}, \dots, \phi_{n,n+1}, \dots, \phi_{nm},$$

so daß

$$\left(\prod_{r=1}^n \prod_{s=m}^{r+1} J(r,s; \phi_{rs}) \right) A = \begin{bmatrix} * & \dots & * \\ & \ddots & \vdots \\ 0 & & * \\ \hline 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix}, \quad (7.19)$$

⁸¹Das etwas seltsame Produkt soll die Reihenfolge der Multiplikation andeuten!

was wieder nichts anderes als eine QR-Zerlegung ist. Was bleibt ist die Bestimmung eines Drehwinkels ϕ_{rs} , $s = r + 1, \dots, m$, $r = 1, \dots, n$, so daß im jeweiligen Schritt

$$\begin{bmatrix} \cos \phi_{rs} & \sin \phi_{rs} \\ -\sin \phi_{rs} & \cos \phi_{rs} \end{bmatrix} \begin{bmatrix} a_{rr} \\ a_{rs} \end{bmatrix} = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}, \quad \alpha^2 = a_{rr}^2 + a_{rs}^2,$$

was sich immer lösen läßt, wie man sich leicht geometrisch vorstellt. Allerdings brauchen wir nicht die Winkel selbst zu bestimmen, es genügen die Werte von deren Sinus und Cosinus! Die bekommt man aber ganz einfach. Setzen wir

$$c_{rs} = \cos \phi_{rs} \quad \text{und} \quad s_{rs} = \sin \phi_{rs},$$

dann erhalten wir aus der zweiten Zeile, daß

$$s_{rs} a_{rr} = c_{rs} a_{rs} \quad \Rightarrow \quad \begin{cases} s_{rs} = \frac{(\text{sign } a_{rr}) |a_{rs}|}{\sqrt{a_{rr}^2 + a_{rs}^2}}, \\ c_{rs} = \frac{(\text{sign } a_{rs}) |a_{rr}|}{\sqrt{a_{rr}^2 + a_{rs}^2}}, \end{cases}$$

wobei $\text{sign } 0 = 1$ gesetzt wird⁸². Bei dieser Wahl von c_{rs} und s_{rs} ergibt sich übrigens

$$\alpha = (\text{sign } a_{rr}) (\text{sign } a_{rs}) \sqrt{a_{rr}^2 + a_{rs}^2}.$$

Übung 7.7 Implementieren Sie die QR-Zerlegung mittels Givens-Rotationen in Matlab oder Octave. ◇

⁸²Aber nur der Vollständigkeit wegen, der Wert -1 wäre genau so gut.

*And now for something completely
different ...*

Monty Python's Flying Circus

Polynome | 8

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 8.1 Eine Funktion der Form

$$p(x) = \sum_{j \in \mathbb{N}_0} a_j x^j, \quad a_j \in \mathbb{K},$$

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

$$\Pi_n = \{p : p \text{ Polynom, } \deg p \leq n\}$$

bezeichnet man den Vektorraum aller Polynome vom Grad höchstens n und mit $\Pi = \mathbb{K}[x]$ den Vektorraum aller Polynome.

Bemerkung 8.2 (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

$$\dim \Pi_n = n + 1, \quad n \in \mathbb{N}_0$$

und die **Monome** $\{1, x, \dots, x^n\}$ 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. Das werden wir aber in diesem Abschnitt praktisch nicht ausnutzen.

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 [43].

Satz 8.3 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

$$\|f - p\|_I := \max_{x \in I} |f(x) - p(x)| < \varepsilon. \quad (8.1)$$

Der Beweis findet sich in Anhang A.3.

8.1 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 der verwendeten Darstellung ab. Im CAGD⁸³ verwendet man nicht nur polynomiale *Funktionen* sondern polynomiale *Kurven*.

Definition 8.4 Eine Abbildung

$$\mathbf{p} = \begin{bmatrix} p_1 \\ \vdots \\ p_d \end{bmatrix} : \mathbb{R} \rightarrow \mathbb{R}^d$$

heißt **polynomiale Kurve** vom (Höchst-) Grad n , falls $p_j \in \Pi_n$, $j = 1, \dots, d$. In diesem Fall schreiben wir auch $\mathbf{p} \in \Pi_n^d$.

Bemerkung 8.5 Jede polynomiale Kurve $\mathbf{p} \in \Pi_n^d$ läßt sich schreiben als

$$\begin{aligned} \mathbf{p}(x) &= \begin{bmatrix} a_{10} + a_{11}x + \dots + a_{1n}x^n \\ \vdots \\ a_{d0} + a_{d1}x + \dots + a_{dn}x^n \end{bmatrix} = \underbrace{\begin{bmatrix} a_{10} \\ \vdots \\ a_{d0} \end{bmatrix}}_{=\mathbf{a}_0} + \underbrace{\begin{bmatrix} a_{11} \\ \vdots \\ a_{d1} \end{bmatrix}}_{=\mathbf{a}_1} x + \dots + \underbrace{\begin{bmatrix} a_{1n} \\ \vdots \\ a_{dn} \end{bmatrix}}_{=\mathbf{a}_n} x^n \\ &= \sum_{j=0}^n \mathbf{a}_j x^j, \end{aligned}$$

wobei $\mathbf{a}_j \in \mathbb{R}^d$, $j = 1, \dots, n$. Damit sind diese Kurven also nichts anderes als „ganz normale“ Polynome mit vektorwertigen Koeffizienten.

⁸³Computer Aided Geometric Design.

```

%% 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 8.1 Horner.m: Das Hornerschema für vektorwertige Polynome.

Proposition 8.6 (Horner-Schema) *Ist*

$$\mathbf{p}(x) = \sum_{j=0}^n \mathbf{a}_j x^j \quad \in \Pi_n^d$$

und setzt man

$$\mathbf{a}^0(x) = \mathbf{a}_n \tag{8.2}$$

$$\mathbf{a}^j(x) = x \mathbf{a}^{j-1}(x) + \mathbf{a}_{n-j}, \quad j = 1, \dots, n, \tag{8.3}$$

dann ist $\mathbf{a}^n(x) = \mathbf{p}(x)$.

Beweis: Schreibe $\mathbf{p}(x)$ als

$$\mathbf{p}(x) = \mathbf{a}_0 + x (\mathbf{a}_1 + x (\dots x (\mathbf{a}_{n-1} + x \mathbf{a}_n) \dots))$$

und werte von innen nach außen aus. □

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 $\mathbf{a}_j = \frac{1}{j!} \mathbf{p}^{(j)}(0)$, $j = 0, \dots, n$, kaum *geometrische* Information über die resultierende Kurve geben und daß das Design mit diesen Koeffizienten praktisch unmöglich ist.

Definition 8.7 Es sei $I = [t_0, t_1] \subset \mathbb{R}$, $-\infty < t_0 < t_1 < \infty$, ein nichtdegeneriertes kompaktes Intervall⁸⁴ und $x \in \mathbb{R}$. Als **baryzentrische Koordinaten**

$$u(x | I) = (u_0(x | I), u_1(x | I))$$

von x bezüglich I definiert man das Paar von Zahlen, das die beiden Bedingungen

$$x = u_0(x | I) t_0 + u_1(x | I) t_1, \quad u_0(x | I) + u_1(x | I) = 1 \quad (8.4)$$

erfüllt.

Bemerkung 8.8 (Baryzentrische Koordinaten)

1. In (8.4) 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

$$u_0(x | I) = \frac{t_1 - x}{t_1 - t_0} \quad \text{und} \quad u_1(x | I) = \frac{x - t_0}{t_1 - t_0}. \quad (8.5)$$

3. Aus (8.5) folgt sofort, daß

$$u_0(t_j | I) = \delta_{0j} \quad \text{und} \quad u_1(t_j | I) = \delta_{1j}, \quad j = 0, 1. \quad (8.6)$$

4. Die baryzentrischen Koordinaten⁸⁵ beschreiben das Verhältnis, in dem der Punkt x das Intervall I teilt, siehe Abb. 8.1.

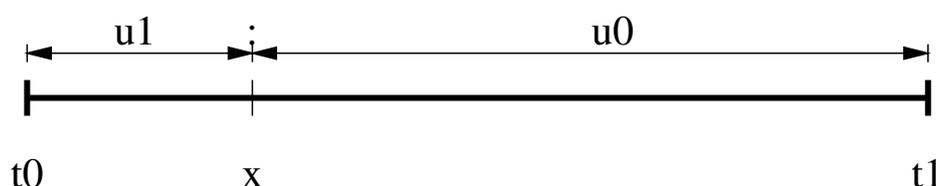


Abbildung 8.1: Baryzentrische Koordinaten als (normiertes) Teilungsverhältnis.

5. Es gilt

$$u_0(x | I), u_1(x | I) \geq 0 \quad \Leftrightarrow \quad x \in I.$$

6. Ist $I = [0, 1]$, dann ist $u_0(x | I) = (1 - x)$ und $u_1(x | I) = x$. Baryzentrische Koordinaten transformieren also das Intervall I in $[0, 1]$.

⁸⁴Also das, was sich der „Mann von der Straße“ so unter einem Intervall vorstellt.

⁸⁵Der Begriff bedeutet ja nichts anderes als „Schwerpunktskoordinaten“.

Mit Hilfe der baryzentrischen Koordinaten können wir ein anderes Auswertungsverfahren für Polynome angeben.

Satz 8.9 Es seien $\mathbf{b}_0, \dots, \mathbf{b}_n \in \mathbb{R}^d$. Für $x \in I$ liefert der **Algorithmus von de Casteljau**⁸⁶,

$$\mathbf{b}_k^0(x) = \mathbf{b}_k, \quad k = 0, \dots, n, \quad (8.7)$$

$$\mathbf{b}_k^{j+1}(x) = u_0(x|I) \mathbf{b}_k^j(x) + u_1(x|I) \mathbf{b}_{k+1}^j(x), \quad k = 0, \dots, n-j-1, \quad (8.8)$$

als Ergebnis den Wert

$$\mathbf{b}_0^n(x) = \sum_{j=0}^n \mathbf{b}_j \underbrace{\binom{n}{j} u_0^{n-j}(x|I) u_1^j(x|I)}_{=: B_j^n(x|I)} \quad (8.9)$$

der **Bézierkurve**⁸⁷

$$B_n \mathbf{b}(x) := B_n(\mathbf{b}_0, \dots, \mathbf{b}_n)(x) = \sum_{j=0}^n \mathbf{b}_j B_j^n(x|I)$$

an der Stelle x .

Definition 8.10 Die Koeffizienten $\mathbf{b} = (\mathbf{b}_0, \dots, \mathbf{b}_n)$ bezeichnet man als **Kontrollpunkte**, die von ihnen gebildete stückweise lineare Funktion als **Kontrollpolygon**.

Die schematische Arbeitsweise des Algorithmus von de Casteljau ist in Abb. 8.2 dargestellt. Die Polynome $B_j^n(x|I)$ sind im Falle $I = [0, 1]$ „alte Bekannte“, nämlich genau die Polynome, mit deren Hilfe wir die **Bernsteinpolynome** im Beweis von Satz 8.3 gebildet haben. Tatsächlich ist ja $B_n f$ auch nichts anderes als eine Bézierkurve, nur eben mit $\mathbf{b}_j = f(j/n)$, $j = 0, \dots, n$. Aus diesem Grund bezeichnet man die Polynome $B_j^n(x|I)$ auch als **Bernstein-Bézier-Basispolynome**.

Übung 8.1 Zeigen Sie, daß jedes stetige Vektorfeld $f : I \rightarrow \mathbb{R}^d$ auf dem kompakten Intervall $I \subset \mathbb{R}$ gleichmäßig durch polynomiale Vektorfelder approximiert werden kann. \diamond

Lemma 8.11 Die Bernstein-Bézier-Basispolynome haben die folgenden Eigenschaften:

1. (Rekursionsformel)

$$B_j^n(x|I) = u_0(x|I) B_j^{n-1}(x|I) + u_1(x|I) B_{j-1}^{n-1}(x|I). \quad (8.10)$$

⁸⁶Paul Faget de Casteljau, Mathematiker, in den 60ern bei Citroën in der CAD-Entwicklung tätig.

⁸⁷Pierre Bézier, † 29.11.1999, Ingenieur, in den 60ern bei Renault in der CAD-Entwicklung tätig.

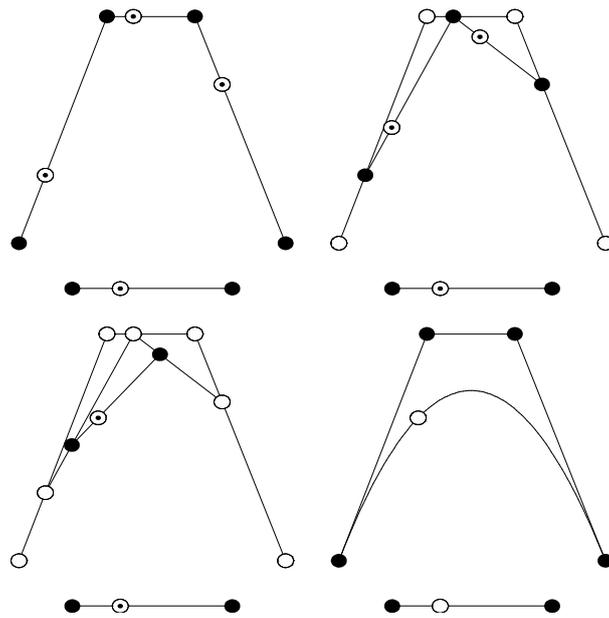


Abbildung 8.2: Der Algorithmus von de Casteljau: Die Teilungsverhältnisse von I bezüglich x werden auf die Streckenzüge übertragen, die die Koeffizienten verbinden.

2. (Teilung der Eins)

$$\sum_{j=0}^n B_j^n(x | I) = 1. \quad (8.11)$$

3. (Nichtnegativität) Für $x \in I$ ist $B_j^n(x | I) \geq 0$.

Beweis: Die erste Identität, (8.10), folgt aus⁸⁸

$$\begin{aligned} B_j^n(x | I) &= \binom{n}{j} u_0^{n-j} u_1^j = \left(\binom{n-1}{j} + \binom{n-1}{j-1} \right) u_0^{n-j} u_1^j \\ &= u_0 \binom{n-1}{j} u_0^{n-1-j} u_1^j + u_1 \binom{n-1}{j-1} u_0^{n-j} u_1^{j-1} \\ &= u_0(x | I) B_j^{n-1}(x | I) + u_1(x | I) B_{j-1}^{n-1}(x | I). \end{aligned}$$

Für (8.11) verwenden wir (8.10) und erhalten mit der Konvention $B_{-1}^n = B_{n+1}^n = 0$, daß

$$\sum_{j=0}^n B_j^n(x | I) = \sum_{j=0}^{n-1} u_0 B_j^{n-1} + \sum_{j=1}^n u_1 B_{j-1}^{n-1} = \underbrace{(u_0 + u_1)}_{=1} \underbrace{\sum_{j=0}^{n-1} B_j^{n-1}}_{=1} = 1$$

⁸⁸Hier und im Folgenden verwenden wir die Kurzschreibweise $u_j = u_j(x | I)$, $x \in I$.

```
%% 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 8.2 DeCasteljau.m: Der Algorithmus von de Casteljau mit Referenzintervall $I = [0, 1]$.

mittels Induktion über n . Die Nichtnegativität folgt direkt aus (8.10). \square

Übung 8.2 Beweisen Sie die kombinatorische Identität

$$\binom{n}{j} = \binom{n-1}{j} + \binom{n-1}{j-1}.$$

\diamond

Beweis von Satz 8.9: Wir zeigen durch Induktion über j , daß

$$\mathbf{b}_k^j(x) = \sum_{r=0}^j \mathbf{b}_{k+r} B_r^j(x | I), \quad k = 0, \dots, n-j, x \in \mathbb{R}, \quad (8.12)$$

was für $j = n$ die Formel (8.9) liefert und für $j = 0$ gerade (8.7), also per Definitionem richtig ist. Für $j \geq 0$ und $k = 0, \dots, n-j-1$ verwenden wir schließlich die Rekursionsformel (8.8), um mit Hilfe der Induktionsannahme

$$\begin{aligned} \mathbf{b}_k^{j+1}(x) &= u_0 \mathbf{b}_k^j(x) + u_1 \mathbf{b}_{k+1}^j = u_0 \sum_{r=0}^j \mathbf{b}_{k+r} B_r^j + u_1 \sum_{r=0}^j \mathbf{b}_{k+r+1} B_r^j \\ &= u_0 \sum_{r=0}^j \mathbf{b}_{k+r} B_r^j + u_1 \sum_{r=1}^{j+1} \mathbf{b}_{k+r} B_{r-1}^j = \sum_{r=0}^{j+1} \mathbf{b}_{k+r} (u_0 B_r^j + u_1 B_{r-1}^j) \\ &= \sum_{r=0}^{j+1} \mathbf{b}_{k+r} B_r^{j+1}(x | I) \end{aligned}$$

zu erhalten, was die Induktion fortsetzt. \square

Bemerkung 8.12 (Eigenschaften der Bézierkurven)

1. („Convex Hull Property“) Da für $x \in I$

$$0 \leq B_j^n(x | I) \quad \text{und} \quad \sum_{j=0}^n B_j^n(x | I) = 1,$$

ist jeder Punkt $B_n \mathbf{b}$ eine **Konvexkombination** von $\mathbf{b}_0, \dots, \mathbf{b}_n$ und damit liegt die Kurve⁸⁹ $B_n \mathbf{b}(I)$ in der konvexen Hülle von $\mathbf{b}_0, \dots, \mathbf{b}_n$. Dies ist beispielsweise für die Schnittbestimmung von Kurven wichtig: Man prüft zuerst einmal, ob sich überhaupt die konvexen Hüllen (also Polygone⁹⁰) schneiden, bevor man ein kompliziertes Verfahren zur Schnittberechnung von Kurven anwirft.

⁸⁹Genauer: ihr Graph.

⁹⁰Zur Bestimmung des Schnitts von Polygonen sind eine Vielzahl von sehr effizienten Algorithmen, Stichwort „Clipping“, verfügbar.

2. („Endpoint Interpolation“) Ist $u_0(x | I) = 1$, dann ist $\mathbf{b}_0^j(x) = \mathbf{b}_0$, $j = 0, \dots, n$, also auch $B_n \mathbf{b}(x) = \mathbf{b}_0$; entsprechend natürlich auch $B_n \mathbf{b}(x) = \mathbf{b}_n$, wenn $u_1(x | I) = 1$. Die Bézierkurven interpolieren also an den Endpunkten des Kontrollpolygons.
3. („Linear Precision“) Liegen alle Kontrollpunkte \mathbf{b}_j , $j = 0, \dots, n$, auf einer Geraden, so ist $B_n \mathbf{b}(I)$ ein Streckenzug auf dieser Geraden: Für alle $x \in \mathbb{R}$ mit $\mathbf{b}_k^j(x)$ und $\mathbf{b}_{k+1}^j(x)$ liegt auch der Zwischenpunkt $\mathbf{b}_k^{j+1}(x)$ auf der Geraden. Allerdings ist im Normalfall die Kurve $B_n \mathbf{b}(x)$ anders parametrisiert als der Streckenzug. Zum Beispiel durchläuft für

$$\mathbf{b}_0 = \mathbf{b}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \mathbf{b}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

die Kurve $B_2 \mathbf{b}$ den Streckenzug „hin und zurück“, siehe Abb. 8.3.

4. Die Form des Kontrollpolygons beschreibt „ungefähr“ die Form der resultierenden Kurve.

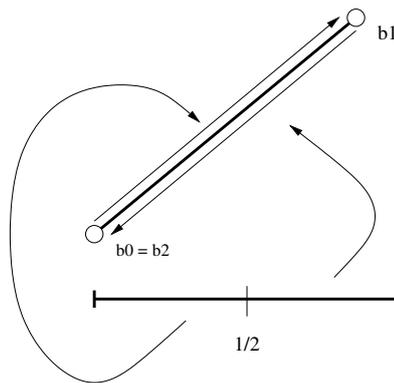


Abbildung 8.3: Eine Bézierkurve, deren Graph ein Streckenzug ist, der aber zweimal durchlaufen wird, das aber nicht ganz – die Kurve erreicht \mathbf{b}_1 nie!

8.2 Interpolation – der Aitken–Neville-Algorithmus

In vielen Anwendungen spielt das folgende **Interpolationsproblem** eine wichtige Rolle:

Gegeben seien Punkte $x_0, \dots, x_n \in I \subset \mathbb{R}$, I ein kompaktes Intervall. Zu vorgegebenen Werten $f_0, \dots, f_n \in \mathbb{R}^d$ finde man eine stetige Funktion $f : I \rightarrow \mathbb{R}$, so daß

$$f(x_j) = f_j, \quad j = 0, \dots, n.$$

Derartige mathematische Aufgaben entstehen oder entstanden zum Beispiel bei der Erstellung ballistischer Tabellen, bzw. bei der Approximation heuristischer (d.h. gemessener) Tabellen⁹¹

Natürlich hat das obige Interpolationsproblem unendlich viele Lösungen. Die „prominenteste“ und „klassischste“ sind sicherlich Polynome.

Satz 8.13 Zu vorgegebenen Stellen⁹² $x_0 < x_1 < \dots < x_n$ und Daten f_0, \dots, f_n gibt es genau ein Polynom $p \in \Pi_n$, so daß

$$p(x_j) = f_j, \quad j = 0, \dots, n. \quad (8.13)$$

Der „Standardbeweis“ für die obige Aussage verwendet die **Lagrange-Darstellung**

$$p(x) = \sum_{j=0}^n f_j \prod_{\substack{k=0 \\ k \neq j}}^n \frac{x - x_k}{x_j - x_k}$$

des Interpolationspolynoms p , die aber numerisch *hochgrading* instabil ist. Wir wollen hier einen anderen, algorithmischen Weg gehen. Dazu schreiben wir

$$I_k^j := [x_k, x_{k+j}], \quad k = 0, \dots, n-j, \quad j = 1, \dots, n.$$

Satz 8.14 Für vorgegebene $x_0 < x_1 < \dots < x_n \in \mathbb{R}$ und $f_0, \dots, f_n \in \mathbb{R}$ liefert der **Aitken–Neville-Algorithmus**⁹³

$$f_k^0(x) = f_k, \quad k = 0, \dots, n \quad (8.14)$$

$$f_k^{j+1}(x) = u_0 \left(x \mid I_k^{j+1} \right) f_k^j(x) + u_1 \left(x \mid I_k^{j+1} \right) f_{k+1}^j(x), \quad k = 0, \dots, n-j-1, \quad (8.15)$$

als Endergebnis ein Polynom $p(x) = f_0^n(x)$ mit der Interpolationseigenschaft

$$p(x_j) = f_j, \quad j = 0, \dots, n. \quad (8.16)$$

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

⁹¹Genau diese Problematik führte I. J. Schoenberg zur „Erfindung“ der Splines während seiner Tätigkeit in den „Ballistic Research Laboratories“, Aberdeen, Maryland, während des zweiten Weltkriegs.

⁹²Diese werden auch gerne als *Knoten* bezeichnet, aber wir werden diesen Namen noch für etwas anderes brauchen.

⁹³Zuerst 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

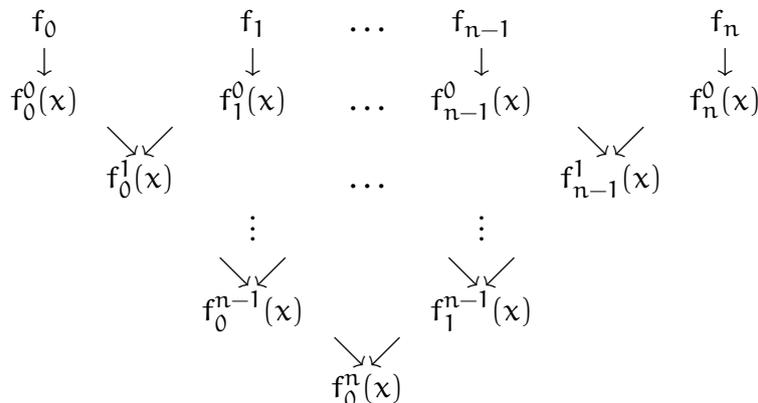
function y = AitkenNeville( f,X,x )
    n = length( f );

    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 8.3 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.

mit den Ausgangsparametern beginnt und dann Zeile für Zeile die Zwischenwerte des Verfahrens auflistet.



Die folgende Aussage beweist nicht nur Satz 8.14, sondern sogar etwas mehr ...

Proposition 8.15 (Eigenschaften von Aitken–Neville)

1. Die Abbildungen $x \mapsto f_k^j(x)$, $k = 0, \dots, n - j$ sind Polynome vom Grad $\leq j$.
2. Für $j = 0, \dots, n$ und $k = 0, \dots, n - j$ gilt:

$$f_k^j(x_m) = f_m, \quad m = k, \dots, k + j, \tag{8.17}$$

insbesondere gilt also (8.16).

Beweis: Eigenschaft 1. folgt sofort aus (8.14), (8.15) und der Tatsache, daß $u_j(\cdot | I) \in \Pi_1$, $j = 1, 2$, für jedes $I \subset \mathbb{R}$.

(8.17) beweisen wir durch Induktion über j , wobei der Fall $j = 0$ genau der Initialisierungsschritt im Algorithmus von Aitken-Neville ist. Sei also (8.17) für ein $j \geq 0$ bewiesen. Dann ist, nach (8.14) und der Induktionsannahme,

$$f_k^{j+1}(x_k) = \underbrace{u_0(x_k | I_k^{j+1})}_{=1} \underbrace{f_k^j(x_k)}_{=f_k} + \underbrace{u_1(x_k | I_k^{j+1})}_{=0} f_{k+1}^j(x_k) = f_k,$$

sowie

$$f_k^{j+1}(x_{k+j+1}) = \underbrace{u_0(x_{k+j+1} | I_k^{j+1})}_{=0} f_k^j(x_{k+j+1}) + \underbrace{u_1(x_{k+j+1} | I_k^{j+1})}_{=1} \underbrace{f_{k+1}^j(x_{k+j+1})}_{=f_{k+j+1}} = f_{k+j+1}.$$

Für $m = k + 1, \dots, k + j$ haben wir schließlich nach Induktionsannahme daß $f_k^j(x_m) = f_{k+1}^j(x_m) = f_m$ und daher

$$\begin{aligned}
 f_k^{j+1}(x_m) &= u_0(x_m | I_k^{j+1}) \underbrace{f_k^j(x_m)}_{=f_m} + u_1(x_m | I_k^{j+1}) \underbrace{f_{k+1}^j(x_m)}_{=f_m} \\
 &= \underbrace{\left(u_0(x_m | I_k^{j+1}) + u_1(x_m | I_k^{j+1}) \right)}_{=1} f_m = f_m
 \end{aligned}$$

also gilt (8.17) auch für $j + 1$, womit der Beweis beendet ist. \square

Beweis von Satz 8.13: 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 $\mathbf{p} \in \Pi_n$ und sei (ohne Einschränkung) $\mathbf{a}_n \neq 0$. Nach dem Satz von Rolle hat dann die Ableitung $\mathbf{p}^{(j)}$ mindestens $n + 1 - j$ Nullstellen, also gibt es (mindestens) ein $x \in \mathbb{R}$, so daß

$$0 = \frac{\mathbf{p}^{(n)}(x)}{n!} = \mathbf{a}_n,$$

was einen Widerspruch liefert. \square

8.3 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 $O(n^2)$, was deutlich mehr als das $O(n)$ des Hornerchemas⁹⁴ ist. Muß man also ein Interpolationspolynom sehr oft auswerten, kann sich eine andere Darstellung lohnen.

Definition 8.16 Es seien $x_0 < x_1 < \dots < x_n$ und $f = [f_0, \dots, f_n] \in \mathbb{R}^{n+1}$.

1. Der **Interpolationsoperator** $L_n : \mathbb{R}^{n+1} \rightarrow \Pi_n$ ordnet jedem Satz f von Eingangsdaten das eindeutige Polynom $L_n f \in \Pi_n$ mit der Eigenschaft

$$L_n f(x_j) = f_j, \quad j = 0, \dots, n.$$

2. Für $f \in C(\mathbb{R})^d$ können wir $L_n f$ ganz einfach erweitern, indem wir f mit $[f(x_0), \dots, f(x_n)]$ identifizieren.
3. Die **Newton–Darstellung** von $L_n f$ hat die Form

$$L_n f(x) = \sum_{j=0}^n \lambda_j(f) (x - x_0) \cdots (x - x_{j-1}),$$

für passende Koeffizienten⁹⁵ $\lambda_j(f) \in \mathbb{R}^d$.

⁹⁴Der Aufwand von de Casteljau ist übrigens $O(n^2)$.

⁹⁵Die wir natürlich noch bestimmen müssen.

Bemerkung 8.17 (Newton–Darstellung)

1. Die Newton–Darstellung erlaubt das einfache Hinzufügen von Punkten: Sind ein weiterer Punkt x_{n+1} und ein weiterer Wert f_{n+1} gegeben, dann ist

$$L_{n+1}f(x) = L_n f(x) + \lambda_{n+1}(f) (x - x_0) \cdots (x - x_n).$$

2. 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, \dots, x_n verschwindet.

Machen wir uns jetzt also an die Konstruktion der Koeffizienten $\lambda_j(f)$ in der Newton–Darstellung. Dazu bemerken wir zuerst, daß uns die Interpolationsbedingungen das Gleichungssystem

$$\sum_{j=0}^n \lambda_j(f) (x_k - x_0) \cdots (x_k - x_{j-1}) = f_k, \quad k = 0, \dots, n,$$

also

$$\begin{bmatrix} 1 & x_0 - x_0 & (x_0 - x_0)(x_0 - x_1) & \cdots & \prod_{j=0}^{n-1} (x_0 - x_j) \\ 1 & x_1 - x_0 & (x_1 - x_0)(x_1 - x_1) & \cdots & \prod_{j=0}^{n-1} (x_1 - x_j) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n - x_0 & (x_n - x_0)(x_n - x_1) & \cdots & \prod_{j=0}^{n-1} (x_n - x_j) \end{bmatrix} \begin{bmatrix} \lambda_0(f) \\ \lambda_1(f) \\ \vdots \\ \lambda_n(f) \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix}.$$

Die Matrix auf der linken Seite ist eine untere Dreiecksmatrix, also ist

$$\begin{bmatrix} 1 & & & \\ 1 & * & & \\ \vdots & \vdots & \ddots & \\ 1 & * & \cdots & * \end{bmatrix} \begin{bmatrix} \lambda_0(f) \\ \lambda_1(f) \\ \vdots \\ \lambda_n(f) \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix},$$

also hängt $\lambda_j(f)$ nur von f_0, \dots, f_j und x_0, \dots, x_j ab und zwar *linear* in f_0, \dots, f_j .

Definition 8.18 Die *dividierte Differenz* $[x_0, \dots, x_n]f$ zu einem Vektor⁹⁶ $f \in \mathbb{R}^{n+1}$ oder einer Funktion $f \in C(\mathbb{R})^d$ ist rekursiv definiert als

$$[x_j]f = f_j = f(x_j), \quad j = 0, \dots, n \quad (8.18)$$

$$[x_0, \dots, x_{j+1}]f = \frac{[x_0, \dots, x_j]f - [x_1, \dots, x_{j+1}]f}{x_0 - x_{j+1}}, \quad j = 0, \dots, n-1. \quad (8.19)$$

```
%% 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 8.4 DivDiff.m: Berechnung der dividierten Differenzen über ein Dreiecksschema aus der Rekursionsformel (8.19).

```

%% 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 8.5 `NewtonKoeff.m`: Naive Berechnung des Koeffizientenvektors für das Interpolationspolynom, jede dividierte Differenz wird separat berechnet.

Satz 8.19 *Es seien $x_0, \dots, x_n \in \mathbb{R}$ alle verschieden und $f \in \mathbb{R}^{d \times n+1}$ oder $f \in C(\mathbb{R})^d$. Dann ist*

$$L_n f(x) = \sum_{j=0}^n [x_0, \dots, x_j] f(x - x_0) \cdots (x - x_{j-1}), \quad x \in \mathbb{R}, \quad (8.20)$$

und außerdem

$$f(x) - L_n f(x) = (x - x_0) \cdots (x - x_n) [x, x_0, \dots, x_n] f. \quad (8.21)$$

Bevor wir uns mit einigen Konsequenzen aus der Newton-Darstellung des Interpolationspolynoms und natürlich dem Beweis von Satz 8.19 beschäftigen, sehen wir uns erst schnell an, wie man so ein Interpolationspolynom mit Matlab oder Octave 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 $[x_0, \dots, x_j] f$ brauchen wir ja alle dividierten Differenzen $[x_0, \dots, x_k] f$, $k < j$. Deswe-

⁹⁶Ob Zeile oder Spalte ist an dieser Stelle völlig egal, aus Implementationssicht ist allerdings eine Zeile eleganter.

```

%% 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 8.6 NewtonKoeff1.m: Effiziente Berechnung des Koeffizientenvektors mit Überschreiben. Beachte: Der Aufwand zur Berechnung aller dividierten Differenzen für das Interpolationspolynom ist genau derselbe wie für die Berechnung des höchsten Koeffizienten $[x_0, \dots, x_n] f$.

gen packen wir alles in ein Schema mit Überschreiben:

$$\begin{array}{c|c|c|c|c|c}
 [x_0] f & \searrow & [x_0] f & \dots & [x_0] f & [x_0] f \\
 [x_1] f & \rightarrow & [x_0, x_1] f & \dots & [x_0, x_1] f & [x_0, x_1] f \\
 [x_2] f & \searrow & [x_1, x_2] f & \dots & [x_0, x_1, x_2] f & [x_0, x_1, x_2] f \\
 \vdots & \searrow & \vdots & & \vdots & \vdots \\
 [x_{n-1}] f & \rightarrow & [x_{n-2}, x_{n-1}] f & \dots & [x_0, \dots, x_{n-1}] f & [x_0, \dots, x_{n-1}] f \\
 [x_n] f & \searrow & [x_{n-1}, x_n] f & \dots & [x_1, \dots, x_n] f & \searrow \\
 & & & & & [x_0, \dots, x_n] f
 \end{array}$$

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 8.7 NHorner.m: Das Hornerschema für die Newtondarstellung eines Polynoms.

den gewünschten Koeffizientenvektor. 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 und einer wichtigen, aber sehr einfach zu beweisenden Eigenschaft der dividierten Differenzen; zumindest, wenn man die Darstellung (8.20) mal hat⁹⁷.

Korollar 8.20 *Die dividierten Differenzen sind symmetrisch in den Argumenten: Für jede Permutation $\sigma : \{0, \dots, n\} \rightarrow \{0, \dots, n\}$ ist*

$$[x_{\sigma(0)}, \dots, x_{\sigma(n)}] f = [x_0, \dots, x_n] f. \quad (8.22)$$

Beweis: Nach (8.20) ist

$$L_n f(x) = [x_0, \dots, x_n] f x^n + q(x), \quad q \in \Pi_{n-1},$$

und da $L_n f$ invariant unter Vertauschung der Interpolationspunkte (bei gleichzeitiger Vertauschung der zu interpolierenden Werte) ist, folgt die Behauptung. \square

Das folgende Resultat zeigt uns, was der Newton Ansatz algorithmisch macht: In jedem Schritt wird versucht, durch geeignete Interpolation des Fehlers⁹⁸ die Genauigkeit

⁹⁷Man kann auch den umgekehrten Weg gehen und die dividierte Differenz als den **Leitkoeffizienten** des Interpolationspolynoms definieren; dann muss man allerdings die Rekursionsformel beweisen.

⁹⁸Ohne die vorher bereits erzielten Interpolationseigenschaften zu zerstören.

des Interpolanten Schritt für Schritt zu erhöhen.

Korollar 8.21 Es seien $x_0, \dots, x_n \in \mathbb{R}$ alle verschieden und $f \in \mathbb{R}^{n+1}$ oder $f \in C(\mathbb{R})$. Dann ist⁹⁹

$$L_n f(x) = \sum_{j=0}^n (f - L_{j-1} f)(x_j) \frac{(x - x_0) \cdots (x - x_{j-1})}{(x_j - x_0) \cdots (x_j - x_{j-1})}. \quad (8.23)$$

Beweis: Nach (8.21) mit $n = j - 1$ und $x = x_j$ ist

$$[x_0, \dots, x_j] f = \frac{(f - L_{j-1} f)(x_j)}{(x_j - x_0) \cdots (x_j - x_{j-1})}, \quad j = 0, \dots, n.$$

□

Beweis von Satz 8.19: Wir beweisen (8.20) und (8.21) simultan durch Induktion über n .

Für $n = 0$ ist $L_0 f(x) = f_0 = f(x_0)$ und

$$f(x) - L_0 f(x) = f(x) - f(x_0) = (x - x_0) \frac{f(x) - f(x_0)}{(x - x_0)} = (x - x_0) [x_0, x] f.$$

Seien also (8.20) und (8.21) für ein $n \geq 0$ bewiesen. Wir setzen

$$\begin{aligned} P_{n+1} f(x) &:= \sum_{j=0}^{n+1} [x_0, \dots, x_j] f (x - x_0) \cdots (x - x_{j-1}) \\ &= P_n f(x) + [x_0, \dots, x_{n+1}] f (x - x_0) \cdots (x - x_n). \end{aligned}$$

Nach der Induktionsannahme erhalten wir aus (8.20) und (8.23)¹⁰⁰, daß

$$P_{n+1} f(x) = L_n f(x) + (f - L_n f)(x_{n+1}) \frac{(x - x_0) \cdots (x - x_n)}{(x_{n+1} - x_0) \cdots (x_{n+1} - x_n)}.$$

Damit ist, für $j = 0, \dots, n$,

$$P_{n+1} f(x_j) = L_n f(x_j) = f_j$$

und außerdem

$$P_{n+1} f(x_{n+1}) = L_n f(x_{n+1}) + f(x_{n+1}) - L_n f(x_{n+1}) = f(x_{n+1}) = f_{n+1},$$

also $P_{n+1} f = L_{n+1} f$ und (8.20) gilt auch für $n + 1$. Andererseits ist mit dem gerade bewiesenen und nochmals der Induktionsannahme

$$\begin{aligned} f(x) - L_{n+1} f(x) &= (f(x) - L_n f(x)) - (L_{n+1} f(x) - L_n f(x)) \\ &= (x - x_0) \cdots (x - x_n) [x, x_0, \dots, x_n] f - (x - x_0) \cdots (x - x_n) [x_0, \dots, x_n, x_{n+1}] f \\ &= (x - x_0) \cdots (x - x_{n+1}) \frac{[x, x_0, \dots, x_n] f - [x_0, \dots, x_n, x_{n+1}] f}{x - x_{n+1}} \\ &= (x - x_0) \cdots (x - x_{n+1}) [x, x_0, \dots, x_{n+1}] f, \end{aligned}$$

⁹⁹Mit $L_{-1}(x) = 0$.

¹⁰⁰Da (8.23) direkt aus (8.21) folgt ist dessen Verwendung zulässig!

nach der Rekursionsformel (8.19). □

Übung 8.3 Zeigen Sie:

$$p \in \Pi_n \quad \Leftrightarrow \quad [x_0, \dots, x_k] p = 0, \quad x_0 < \dots < x_k, \quad k > n.$$

◇

Schließlich noch eine Aussage über den Interpolationsfehler für hinreichend oft differenzierbare Funktionen.

Satz 8.22 Seien x_0, \dots, x_n alle verschieden und $f \in C^{n+1}(\mathbb{R})$. Dann gibt es für jedes $x \in \mathbb{R}$ ein $\xi \in [x_0, \dots, x_n, x]$, der konvexen Hülle von x_0, \dots, x_n, x ¹⁰¹, so daß

$$f(x) - L_n f(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} (x - x_0) \cdots (x - x_n). \quad (8.24)$$

Dann liefern uns die Fehlerformeln (8.24) und (8.21) auch noch eine Aussage über die dividierten Differenzen differenzierbarer Funktionen.

Korollar 8.23 Es sei $f \in C^n(\mathbb{R})$. Zu beliebigen verschiedenen Punkten $x_0, \dots, x_n \in \mathbb{R}$ gibt es ein $\xi \in [x_0, \dots, x_n]$, so daß

$$[x_0, \dots, x_n] f = \frac{f^{(n)}(\xi)}{n!}.$$

Beweis von Satz 8.22: Für $x \notin \{x_0, \dots, x_n\}$ setzen wir

$$\alpha = \frac{f(x) - L_n f(x)}{(x - x_0) \cdots (x - x_n)},$$

definieren

$$g(y) = f(y) - L_n f(y) - \alpha (y - x_0) \cdots (y - x_n), \quad y \in \mathbb{R},$$

und bemerken, daß $g(x_j) = 0$, $j = 0, \dots, n$, sowie $g(x) = 0$. Nach dem Satz von Rolle hat also $g^{(j)}$ mindestens $n + 2 - j$ Nullstellen im Intervall $[x_0, \dots, x_n, x]$, $j = 0, \dots, n + 1$, es gibt also mindestens ein $\xi \in [x_0, \dots, x_n, x]$, so daß

$$\begin{aligned} 0 &= g_\alpha^{(n+1)}(\xi) = f^{(n+1)}(\xi) - \underbrace{(L_n f(x))^{(n+1)}}_{=0} - \alpha \underbrace{\frac{d^{n+1}}{dx^{n+1}} (x - x_0) \cdots (x - x_n)}_{=(n+1)!} \\ &= f^{(n+1)}(\xi) - \alpha (n+1)!, \end{aligned}$$

also ist $\alpha = f^{(n+1)}(\xi) / (n+1)!$. □

¹⁰¹Alternativ: das kleinste Intervall, das x_0, \dots, x_n und x enthält.

8.4 Approximationsgüte von Interpolationspolynomen

Wir beginnen mit einer (trivialen) Folgerung aus Satz 8.22.

Korollar 8.24 Für $f \in C^{n+1}(\mathbb{R})$ und $x_0, \dots, x_n \in I \subset \mathbb{R}$ ist

$$\|f - L_n f\|_I = \max_{x \in I} |f(x) - L_n f(x)| \leq \frac{\|f^{(n+1)}\|_I}{(n+1)!} \max_{x \in I} |x - x_0| \cdots |x - x_n|. \quad (8.25)$$

Wollen wir nun, daß der Interpolant $L_n f$ die Funktion f gut approximiert¹⁰², dann haben wir nur eine Wahl: Wir müssen die Interpolationspunkte x_0, \dots, 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 8.25 Es bezeichne $\mathcal{P}_n \subset \Pi_n$ die Menge aller Polynome vom Grad n mit monomialem Leitterm bzw. Leitkoeffizient 1:

$$p \in \mathcal{P}_n \quad \Leftrightarrow \quad p(x) = x^n + q(x), \quad q \in \Pi_{n-1}.$$

Ist also p^* das Polynom, für das

$$\|p^*\|_I = \min_{p \in \mathcal{P}_n} \|p\|_I$$

gilt, dann sind die Nullstellen¹⁰³ von p^* optimale Interpolationspunkte.

Unser Ziel wird es nun sein, für alle $n \in \mathbb{N}_0$ so ein optimales p^* zu finden.

Definition 8.26 Das n -te Tschebyscheff-Polynom¹⁰⁴ T_n , $n \in \mathbb{N}$, ist definiert als

$$T_n(x) = \cos(n \arccos x).$$

Und, kaum zu glauben, aber wahr, die Tschebyscheffpolynome sind die Lösung unseres Problems.

¹⁰²Das wäre ja wohl das, was man als „gute Rekonstruktion“ bezeichnen würde.

¹⁰³Man sollte aber schon nachweisen, daß diese Nullstellen auch alle einfach und reell sind!

¹⁰⁴Es gibt eine Vielzahl von Transkriptionen des Namen „Tschebyscheff“, z.B. auch Chebychev, Chebychov.

Satz 8.27 Die Polynome $2^{1-n}T_n$ gehören zu \mathcal{P}_n , $n \geq 1$ ¹⁰⁵, und es gilt

$$\left\| 2^{1-n}T_n \right\|_{[-1,1]} = \min_{p \in \mathcal{P}_n} \| p \|_{[-1,1]}. \quad (8.26)$$

Die Nullstellen von T_{n+1} ¹⁰⁶, $n \in \mathbb{N}_0$, sind alle einfach und reell und damit optimale Interpolationspunkte für Π_n .

Korollar 8.28 Sei L_n^* der Interpolationsoperator an den Nullstellen von T_{n+1} . Dann ist

$$\| f - L_n^* f \|_{[-1,1]} \leq \frac{\| f^{(n+1)} \|_{[-1,1]}}{2^n (n+1)!}. \quad (8.27)$$

Um einzusehen, daß tatsächlich $T_n \in \Pi_n$ ist, brauchen wir etwas mehr Information über die Tschebyscheff-Polynome.

Lemma 8.29 Die Tschebyscheff-Polynome T_n , $n \in \mathbb{N}$, erfüllen die Rekursionsformel

$$T_0(x) = 1, \quad (8.28)$$

$$T_1(x) = x, \quad (8.29)$$

$$T_{n+1}(x) = 2x T_n(x) - T_{n-1}(x), \quad n \in \mathbb{N}. \quad (8.30)$$

Beweis: Die Anfangsbedingungen (8.28) und (8.29) 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**

$$\cos(n+1)\theta = \cos n\theta \cos \theta + \sin n\theta \sin \theta$$

$$\cos(n-1)\theta = \cos n\theta \cos \theta - \sin n\theta \sin \theta$$

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 (8.30). □

Als nächstes Information über die Nullstellen.

Proposition 8.30 Die Nullstellen von T_n sind

$$x_j = \cos \frac{2j-1}{2n} \pi, \quad j = 1, \dots, n. \quad (8.31)$$

¹⁰⁵Der Fall $n = 0$ verlangt den Faktor 1 und nicht $\frac{1}{2}$, siehe (8.28); andererseits hat T_0 auch keine Nullstelle und ist daher ohnehin zur Bestimmung von Interpolationsknoten ungeeignet.

¹⁰⁶Siehe (8.31).

```

%% 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 8.8 TschebNodes.m: Knotenvektor der Tschbyscheffpolynome.

Außerdem sind die Nullstellen von T_n'

$$x_j' = \cos \frac{j}{n} \pi, \quad j = 1, \dots, n-1, \quad (8.32)$$

und es ist¹⁰⁷, mit $x_0' = -1$, $x_n' = 1$,

$$T_n(x_j') = (-1)^{n+j}, \quad j = 0, \dots, n. \quad (8.33)$$

Also ist insbesondere $\|T_n\|_{[-1,1]} = 1$.

Beweis: Es ist

$$T_n(x_j) = \cos \left(n \arccos \cos \frac{2j-1}{2n} \pi \right) = \cos \frac{2j-1}{2} \pi = 0, \quad j = 1, \dots, n,$$

und mehr als n Nullstellen kann T_n ja nun mal nicht haben, denn $T_n(x) = 2^{n-1}x^n + \dots \neq 0$ nach (8.30). Deswegen sind die Nullstellen auch alle einfach. Außerdem ist

$$T_n'(x) = \frac{n}{\sqrt{1-x^2}} \sin(n \arccos x)$$

und da $x_j' \neq -1, 1$ und $\sin j\pi = 0$, $j \in \mathbb{N}_0$, ist auch $T_n'(x_j) = 0$, $j = 1, \dots, n-1$. Also sind die potentiellen Extrempunkte von T_n auf dem Intervall $[-1, 1]$ die Punkte

$$-1, x_1', \dots, x_{n-1}', 1.$$

¹⁰⁷Wenn man genau ist: für $n \geq 1$.

Aus (8.30) erhalten wir sofort, daß $T_n(-1) = (-1)^n$ und $T_n(1) = 1$ und darüberhinaus ist

$$T_n(x'_j) = \cos \frac{j}{n} \pi = (-1)^j,$$

woraus (8.32) folgt. □

Beweis von Satz 8.27: Daß T_n einfache, reelle Nullstellen hat, wurde ja in Proposition 8.30 bewiesen. Daß $2^{1-n}T_n \in \mathcal{P}_n$ folgt aus der Rekursionsformel (8.30). Außerdem wissen wir, daß $\|2^{1-n}T_n\|_{[-1,1]} = 2^{1-n}$. Nehmen wir also an, es gäbe ein $p \in \mathcal{P}_n$, so daß

$$\|p\|_{[-1,1]} < 2^{1-n}$$

und setzen wir $q = 2^{1-n}T_n - p$. Da $2^{1-n}T_n, p \in \mathcal{P}_n$, ist $q \in \Pi_{n-1}$. An den Stellen x'_j , $j = 0, \dots, n$ ¹⁰⁸, ist dann

$$q(x'_j) = (-1)^j 2^{1-n} - p(x_j) =: (-1)^j \mu_j, \quad j = 0, \dots, n,$$

wobei $\mu_j > 0$, $j = 0, \dots, n$. Damit muß aber q zwischen x'_j und x'_{j+1} , $j = 0, \dots, n-1$, 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

$$2^{1-n} > \|p\|_{[-1,1]} = \|2^{1-n}T_n\|_{[-1,1]} = 2^{1-n}$$

zum Widerspruch. □

Tatsächlich liefern die Tschebyscheffknoten *entscheidend* bessere Interpolanten als beispielsweise gleichverteilte Knoten.

Beispiel 8.31 Wir betrachten die geschlossene polynomiale Kurve, die durch die Punkte (Matlab-Vektor f)

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

auf dem Rand des Einheitsquadrats geht (siehe Abb. 8.4). Als Parameterbereich wählen wir $[-1, 1]$ und interpolieren einmal an gleichverteilten 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. 8.5 zeigt, sehr unterschiedlich.

¹⁰⁸Wir nehmen also die Grenzen ± 1 mit dazu.

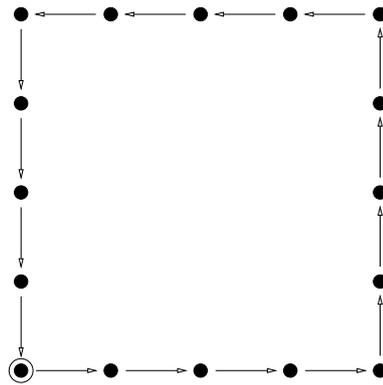


Abbildung 8.4: Interpolationsdaten für die Kurve. Frei ist noch die Wahl der zugehörigen Parameterwerte.

8.5 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. 8.6 zeigt. Allerdings sieht man dort auch, daß es so richtig schlimm erst außerhalb der konvexen Hülle der Interpolationspunkte wird, aber auch in deren Innenen wird die Oszillation für wachsenden Grad immer heftiger

Noch schlimmer ist aber die Empfindlichkeit polynomialer Interpolation gegen Störungen. Dazu ein Beispiel.

Beispiel 8.32 Wir betrachten die Parabel¹⁰⁹ $p(x) = x(1 - x)$ und interpolieren an den 21 gleichverteilten Punkten $0, \frac{1}{20}, \dots, 1$. Das Interpolationspolynom ist auch kaum von der Parabel selbst zu unterscheiden (Abb. 8.7). Stören wir aber jeden Datenwert zufällig um $< 0.1\%$,

¹⁰⁹Ein Schuft ist, wer an Geschößbahnen dabei denkt.

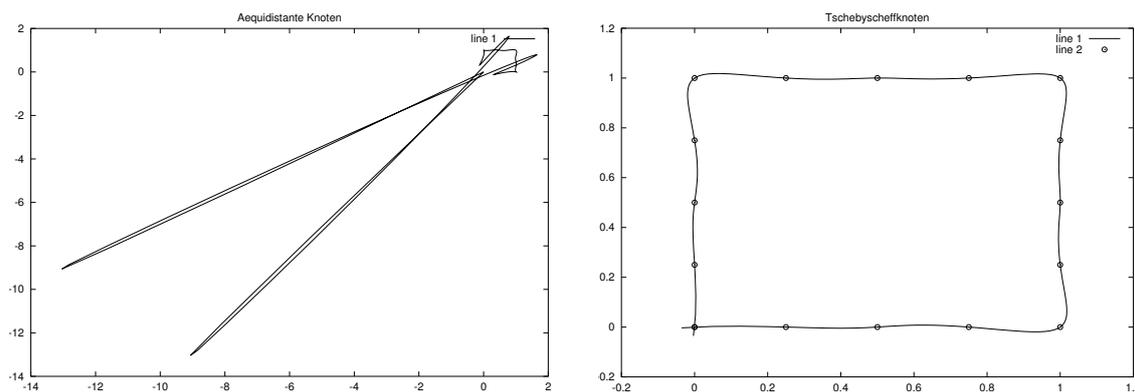


Abbildung 8.5: Interpolation der „Quadratkurve“, einmal mit gleichverteilten Parameterwerten für die Interpolationspunkte, einmal unter Verwendung der Tschebyscheffknoten.

dann erhalten wir bereits eine ziemlich unögliche „Flugbahn“ (Abb. 8.7), stören wir gar den Scheitelpunkt um 1%, so wird der Interpolant endgültig wüst, siehe Abb. 8.8. Auf der anderen Seite zeigt Abb. 8.9, 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 Nullstellen¹¹⁰!

¹¹⁰Wohl aber T_n und T_{2n} – was auch in manchen Verfahren ausgenutzt wird.

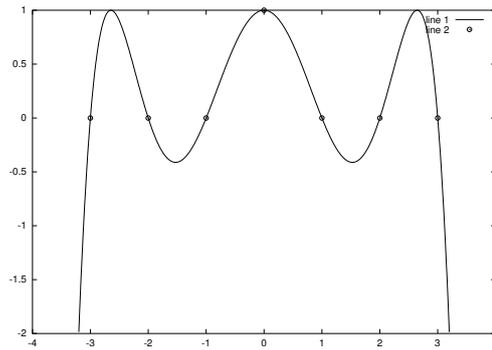


Abbildung 8.6: Interpolationspolynom, das den Wert 0 an $\pm 1, \pm 2$ und 1 an der Stelle 0 interpoliert. Nicht ganz das, was man gerne hätte.

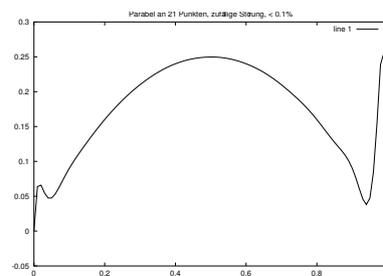
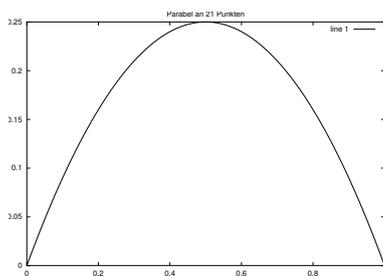


Abbildung 8.7: Interpolanten vom Grad 20 für die Originaldaten und leicht ($< 0.1\%$) gestörte Werte.

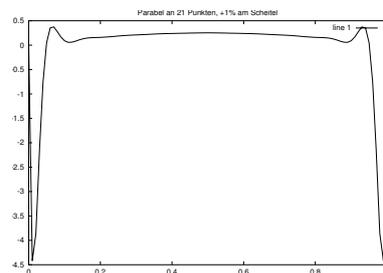
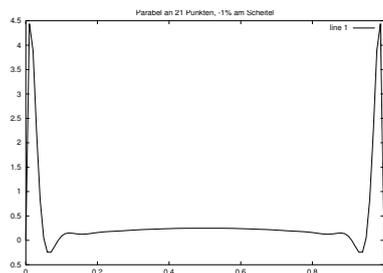


Abbildung 8.8: Interpolanten vom Grad 20 für Störung des Scheitelpunkts um $\pm 1\%$.

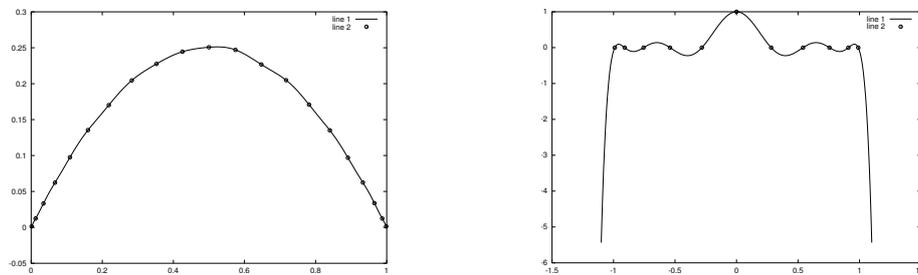


Abbildung 8.9: Zufällige Störungen an *allen* Interpolationspunkten um $< 1\%$ tun bei Verwendung der Tschebyscheffknoten bei weitem nicht so weh. Auch die Oszillation ist nicht so groß, aber dafür wird das Gefälle außerhalb von $[-1, 1]$ umso steiler.

*Im Wort Gelehrter steckt nur der Begriff,
daß man ihn vieles gelehrt, aber nicht,
daß er auch etwas gelernt hat.*

Lichtenberg

Splines | 9

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 9.1 Seien $n \geq m \geq 0$ ¹¹¹.

1. Eine Punktmenge $T = T_{m,n} = \{t_0, \dots, t_{n+m+1}\} \subset \mathbb{R}$ heißt **Knotenfolge** der Ordnung m , falls

$$t_0 \leq t_1 \leq \dots \leq t_n \leq \dots \leq t_{n+m+1} \quad (9.1)$$

und

$$t_j < t_{j+m+1}, \quad j = 0, \dots, n. \quad (9.2)$$

2. Die **Knoten** t_0, \dots, t_m bzw. $t_{n+1}, \dots, t_{n+m+1}$ heißen *linke* bzw. *rechte Randknoten*, die Knoten t_{m+1}, \dots, t_n heißen **innere Knoten**.

3. Dasjenige $\mu > 0$ ¹¹², für das

$$t_{j-1} < t_j = \dots = t_{j+\mu-1} < t_{j+\mu}$$

gilt, heißt **Vielfachheit** des Knotens $t_j = \dots = t_{j+\mu-1}$.

4. Das Intervall I_j^k ist definiert als

$$I_j^k = [t_j, t_{j+k}], \quad j = 0, \dots, n, \quad k = 1, \dots, m.$$

¹¹¹Das ist *formal* ausreichend. Die „Vorstellung“ ist allerdings, daß $n \gg m$ und m *sehr* moderat (etwa $m = 3$) ist.

¹¹²Wie „multiplicity“

9.1 Der Algorithmus von de Boor

Als nächstes definieren wir uns auf *algorithmische Weise* die Kurven, die uns interessieren werden.

Algorithmus 9.2 (de Boor) Gegeben: Knotenfolge $T = T_{m,n}$, Kontrollpunkte $\mathbf{d}_0, \dots, \mathbf{d}_n \in \mathbb{R}^d$ und $x \in [t_m, \dots, t_{n+1}]$.

1. Bestimme $r \in \{m, \dots, n\}$ so daß $x \in [t_r, t_{r+1}]$.¹¹³

2. Setze

$$\mathbf{d}_k^0(x) = \mathbf{d}_k, \quad k = r - m, \dots, r.$$

3. Für $j = 1, \dots, m$ berechne

$$\mathbf{d}_k^j(x) = u_0(x|I_k^{m-j+1}) \mathbf{d}_{k-1}^{j-1}(x) + u_1(x|I_k^{m-j+1}) \mathbf{d}_k^{j-1}(x), \quad k = r - m + j, \dots, r. \quad (9.3)$$

4. Ergebnis: $N_{m,T} \mathbf{d}(x) := \mathbf{d}_r^m(x)$.

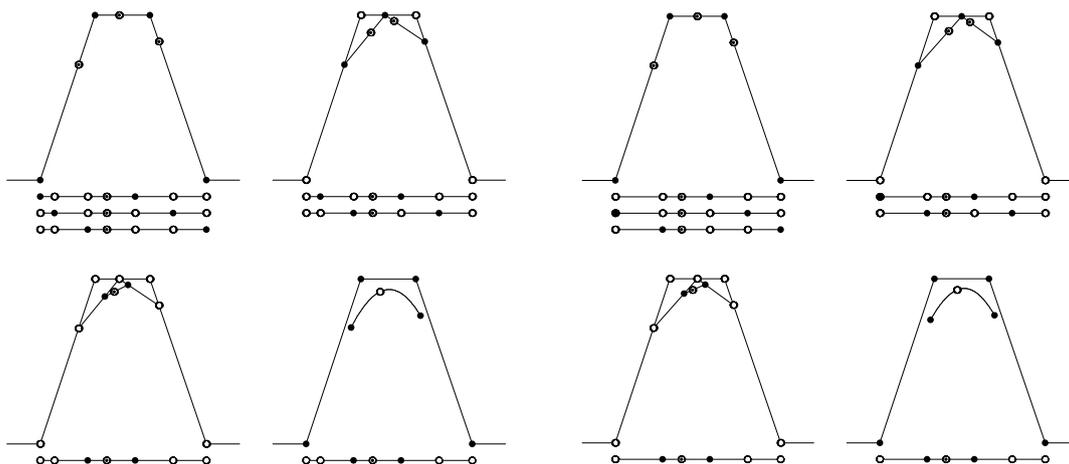


Abbildung 9.1: Der Algorithmus von de Boor für $m = 3$ und einen einfachen sowie einen doppelten Knoten.

Abbildung 9.1 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.

¹¹³Ein solches r existiert *nicht*, wenn $t_{n+1} = \dots = t_{n+m+1}$ 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 9.1 deBoor.m: Der Algorithmus von de Boor mit Überschreiben des Koeffizientenvektors.

Bemerkung 9.3 Der Index r , der durch die Forderung $x \in [t_r, t_{r+1})$ in Algorithmus 9.2 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

$$\mathbf{d}_k^0(x) = \mathbf{d}_k, \quad k = 0, \dots, n.$$

2. Für $j = 1, \dots, m$ berechne

$$\mathbf{d}_k^j(x) = u_0(x|I_k^{m-j+1}) \mathbf{d}_{k-1}^{j-1}(x) + u_1(x|I_k^{m-j+1}) \mathbf{d}_k^{j-1}(x), \quad k = j, \dots, n. \quad (9.4)$$

3. Wähle die r -te Komponente aus dem Ergebnisvektor $(\mathbf{d}_k^m(x) : k = m, \dots, n)$.

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 9.4 Die Kurve $N_{m,T} \mathbf{d} : [t_m, t_{n+1}]$ heißt **Splinekurve vom Grad m** ¹¹⁴.

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 Randes¹¹⁵.
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 $j = 1, \dots, m$ und $k = r - m + j, \dots, r = j, \dots, m$ ¹¹⁶ das Intervall

$$I_k^{m-j+1} = [t_k, t_{k+m-j+1}] = [t_m, t_{k+m-j+1}]$$

zu betrachten¹¹⁷ und somit ist

$$u_0(x|I_k^{m-j+1}) = u_0(t_m | [t_m, t_{k+m-j+1}]) = 1, \quad k = r - m + j, \dots, r, \quad j = 1, \dots, m,$$

also auch

$$\mathbf{d}_{r-m+j}^j(x) = \mathbf{d}_j^j(x) = \mathbf{d}_0,$$

insbesondere ist also $\mathbf{d}_0 = \mathbf{d}_m^m(t_m) = N_{m,T}(t_m)$. Da dasselbe für den rechten Rand gemacht werden kann, besitzen also die Splinekurven die Eigenschaft der *Endpunktinterpolation*, wenn wir $(m+1)$ -fache Randknoten verwenden.

¹¹⁴Oder 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.

¹¹⁵Genauer: auf den beiden Intervallen $[t_m, t_{m+1}]$ und $[t_n, t_{n+1}]$

¹¹⁶Wegen $r = m$

¹¹⁷Schließlich ist $t_j = \dots = t_m, j = 1, \dots, m$.

3. Dies gilt natürlich nicht nur für Randknoten: An jedem m -fachen inneren Knoten $t_j = \dots = t_{j+m-1}$ wird der entsprechende Koeffizient d_{j-m} interpoliert, d.h. $N_{m,T}d(t_j) = d_{j-m}$. An einem $m + 1$ -fachen Knoten werden also sogar d_{j-m} und d_{j-m+1} interpoliert, die Kurve wird dort also normalerweise nicht mehr stetig sein.

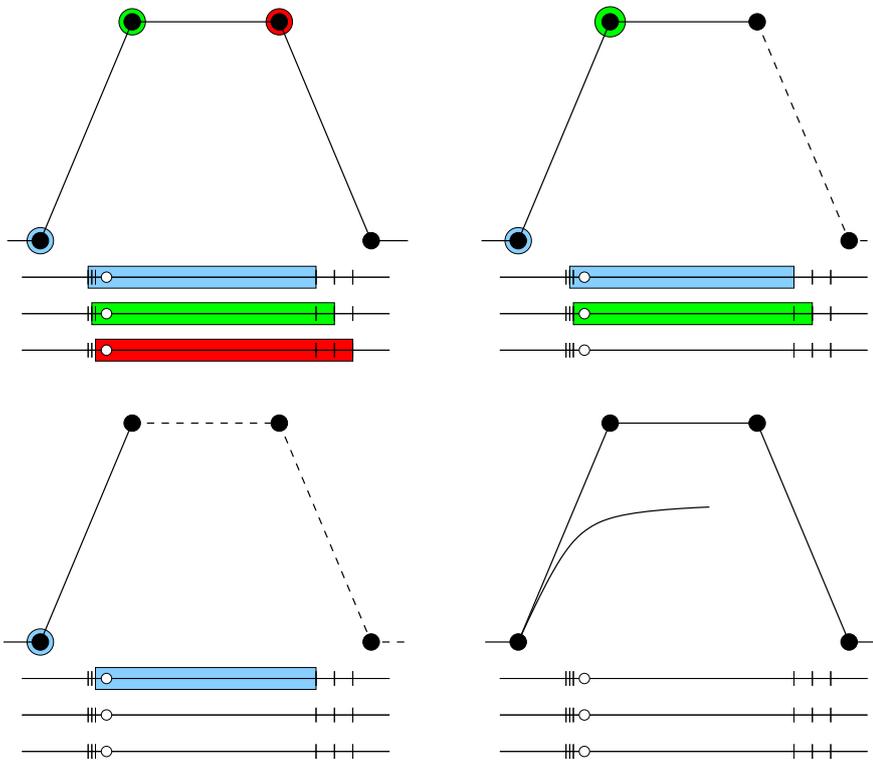


Abbildung 9.2: Der de Boor-Algorithmus an einem m -fachen Knoten t_j (nur „auseinandergezogen“ gezeichnet), für $m = 3$. Die Zwischenpunkte in jedem Schritt sind die Kontrollpunkte und am Schluß bleibt nur d_{j-m} übrig.

4. Spezialisieren wir noch ein bißchen: Ist obendrein noch $m = n$, haben wir also

$$t_0 = \dots = t_n < t_{n+1} = \dots = t_{2n+1},$$

dann ist der Algorithmus von de Boor nichts anderes als der Algorithmus von de Casteljau und es ist

$$N_{n,T}d = B_n d.$$

5. Für alle $x \in (t_r, t_{r+1})$ ¹¹⁸ werden *dieselben* Bezugsintervalle I_k^{m-j+1} verwendet und damit ist $N_{m,T}d|_{(t_r, t_{r+1})}$ ein Produkt von m affinen Funktionen, also ein Polynom vom Grad $\leq m$. Anders gesagt:

¹¹⁸Achtung: Ist $t_r = t_{r+1}$, dann ist $(t_r, t_{r+1}) = \emptyset$!

Splinekurven sind stückweise Polynome!

6. Da

$$N_{m,T}(\alpha \mathbf{d} + \beta \mathbf{d}') = \alpha N_{m,T} \mathbf{d} + \beta N_{m,T} \mathbf{d}'$$

können wir \mathbf{d} als

$$\mathbf{d} = \sum_{j=0}^n \mathbf{d}_j \delta_j, \quad \delta_j = (0, \dots, 0, \underset{\substack{\uparrow \\ j}}{1}, 0, \dots, 0).$$

und damit die Splinekurve als Linearkombination

$$N_{m,T} \mathbf{d}(x) = \sum_{j=0}^n \mathbf{d}_j N_j^m(x|T). \quad (9.5)$$

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 9.5 Die Funktion $N_j^m(\cdot|T)$, aus (9.5) heißt *j-ter B-Spline vom Grad m bezüglich der Knotenfolge T* .

9.2 Splines und Interpolation

Im Rahmen 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 9.6 Es sei $T = T_{m,n}$ eine Knotenfolge und es seien $x_0 < x_1 < \dots < x_n \in \mathbb{R}$.

1. (*Schoenberg-Whitney*) Das Interpolationsproblem

$$N_{m,T} \mathbf{d}(x_j) = f_j, \quad j = 0, \dots, n, \quad (9.6)$$

ist genau dann für alle $f \in \mathbb{R}^{d \times n+1}$ eindeutig lösbar, wenn

$$t_j < x_j < t_{j+m+1}, \quad j = 0, \dots, n. \quad (9.7)$$

2. Ist (9.7) erfüllt, dann ist die **Kollokationsmatrix**

$$[N_k^m(x_j | T) : j, k = 0, \dots, n] \quad (9.8)$$

(m, m)-bandiert und total nichtnegativ.

Bemerkung 9.7

1. Die Bedingung (9.7), die die Knoten und die Interpolationspunkte zueinander 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 (9.7) zu

$$x_{j-m-1} < t_j < x_j, \quad j = m + 1, \dots, n, \quad (9.9)$$

zusammenfasst, was gerade Bedingungen an die Lage der inneren Knoten liefert, siehe Definition 9.1.

2. Zusammen mit (9.7) kann $x_j \in (t_k, t_{k+m+1})$ aber nur dann gelten, wenn

$$j \leq k + m \quad \text{und} \quad j + m \leq k \quad \Rightarrow \quad k - m \leq j \leq k + m$$

ist. Da, wie wir in Lemma 9.14 herausfinden werden, der B-Spline $N_k^m(\cdot | T)$ aber nur im offenen Intervall (t_k, t_{k+m+1}) von Null verschieden¹¹⁹ ist, liefert diese Beobachtung auch bereits die (m, m) -Bandiertheit der **Kollokationsmatrix** (9.8).

Bevor wir uns daran machen, uns die Splines genauer anzusehen, erst einmal ein einfaches Beispiel, das Satz 9.6 ein bißchen illustriert.

Beispiel 9.8 Wir setzen $m = 1$ und betrachten eine Knotenfolge $T = T_{n,1} = t_0 < \dots < t_{n+2}$ ¹²⁰.

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 \mathbf{d}_{k-1} und \mathbf{d}_k zu kombinieren, $k = 1, \dots, n$. Damit ist aber $N_{1,T} \mathbf{d}$ nichts anderes als die stückweise lineare Funktion mit „Bruchstellen“ an t_1, \dots, t_n , die die Punkte $\mathbf{d}_0, \dots, \mathbf{d}_n$ verbindet.
2. Damit können wir $N_{1,T} \mathbf{d}$ als

$$N_{1,T} \mathbf{d} = \sum_{j=0}^n \mathbf{d}_j N_j^1(\cdot | T)$$

schreiben, wobei

$$N_j^1(x|T) = \begin{cases} 0 & x < t_j, \\ \frac{x-t_j}{t_{j+1}-t_j} & x \in [t_j, t_{j+1}], \\ \frac{t_{j+2}-x}{t_{j+2}-t_{j+1}} & x \in [t_{j+1}, t_{j+2}], \\ 0 & x > t_{j+2}. \end{cases}$$

¹¹⁹Und sogar strikt positiv!

¹²⁰Es sind jetzt also *alle* Knoten, auch die Randknoten, *einfach* – aber nur um der Einfachheit willen.

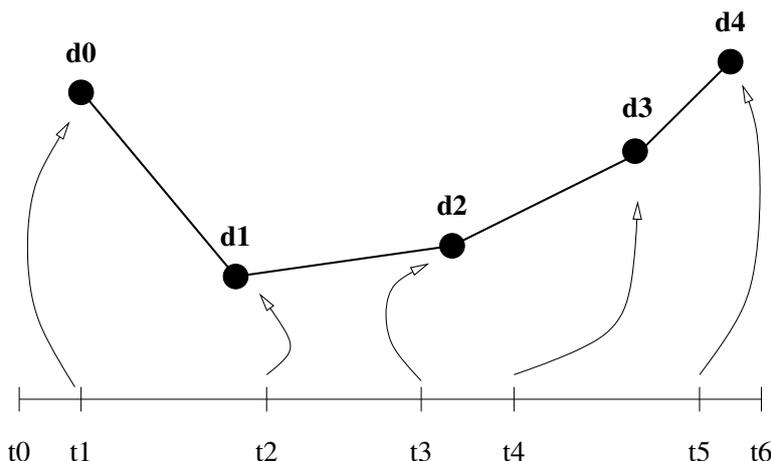


Abbildung 9.3: Die lineare Splinekurve (Ordnung $m = 1$).

3. Was bedeutet nun die Schoenberg–Whitney–Bedingung (9.7)? 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 Kollokationsmatrix eine Nullzeile, wäre nicht invertierbar und damit das Interpolationsproblem unlösbar.

Umgekehrt garantiert (9.7) 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 (9.7) 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.

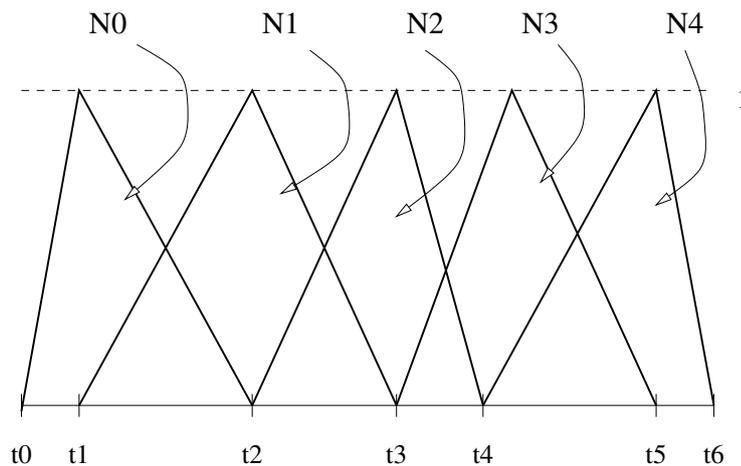


Abbildung 9.4: Die zugehörigen vier B-Splines zur Splinekurve aus Abb 9.3.

- 5. Es gibt noch einen besonders einfachen Fall: $x_j = t_{j+1}$. Dann ist die Kollokationsmatrix sogar die $(0,0)$ -bandierte Einheitsmatrix und $\mathbf{d}_j = \mathbf{f}_j$, aber so einfach ist's halt nicht immer.

Als Folgerung aus dem Satz von Schoenberg–Whitney können wir einen besonderen Splineinterpolanten herleiten.

Satz 9.9 Sei $m = 2r + 1 \in \mathbb{N}$ und $T = T_{m,n}$ eine Knotenfolge mit einfachen Knoten. Dann gibt es zu vorgegebenen Werten $f_j, j = 0, \dots, n - m + 1$, einen Spline $N_{m,T} \mathbf{d}$, so daß¹²¹

$$N_{m,T} \mathbf{d}(t_{m+j}) = f_j, \quad j = 0, \dots, n - m + 1, \tag{9.10}$$

$$N_{m,T}^{(k)} \mathbf{d}(t_\ell) = 0, \quad k = r + 1, \dots, 2r, \ell = m, n + 1. \tag{9.11}$$

Ist $n \geq m + r$, dann ist dieser interpolierende Spline eindeutig.

Definition 9.10 Der¹²² Spline, der die Bedingungen (9.10) und (9.11) erfüllt, heißt **natürlicher Splineinterpolant** an den Knoten t_m, \dots, t_{n+1} .

Beweis: Nach dem Satz von Schoenberg–Whitney, Satz 9.6, ist bei einfachen Knoten das Splineinterpolationsproblem für

$$x_j = t_{j+r+1} \in (t_j, t_{j+2r+1}), \quad j = 0, \dots, n,$$

eindeutig lösbar. Dabei verwenden wir, zusätzlich zu den $n + 2 - m = n + 1 - 2r$ Bedingungen aus (9.10) noch Interpolationsbedingungen an den „letzten“ r linken und

¹²¹Der Spline interpoliert als an den inneren Knoten und den innersten Randknoten.

¹²²Für $n \geq m + r$ ist es „der“, sonst nur „ein“ ...

den „ersten“ r rechten Randknoten; insgesamt sind ja auf jeder Seite $m + 1 = 2r + 2 > r$ Randknoten. Seien nun s_{-r}, \dots, s_r die Lösungen des Interpolationsproblems, das man erhält, wenn man zusätzlich zu (9.10) noch

$$s_j(x_k) = \begin{cases} 1 & j < 0 \text{ und } k = r - j, \\ 1 & j > 0 \text{ und } k = n + j - r, \\ 0 & \text{sonst,} \end{cases} \quad k = 0, \dots, r-1, n+1, \dots, n+r,$$

fordert. Alle diese Splines interpolieren an den Punkten t_m, \dots, t_{n+1} , aber haben ein unterschiedliches Verhalten an den $2r$ zusätzlich eingeführten Punkten t_{r+1}, \dots, t_{m-1} bzw. $t_{n+2}, \dots, t_{n+r+1}$: s_0 verschwindet dort, während s_{-1}, \dots, s_{-r} jeweils an einem der linken „Zusatzknoten“, s_1, \dots, s_r an einem der rechten „Zusatzknoten“ den Wert 1 annehmen, siehe Abb. 9.5.

Diese $m = 2r + 1$ Splines sind $\neq 0$ und linear unabhängig. Betrachten wir nun das lineare Gleichungssystem für a_{-r}, \dots, a_r , das gegeben ist durch

$$\sum_{j=-r}^r a_j s_j^{(k)}(t_\ell) = 0, \quad k = r+1, \dots, 2r, \ell = m, n+1,$$

dann sind dies $2r$ Gleichungen in den $2r + 1$ Unbekannten a_{-r}, \dots, a_r , und es gibt immer (mindestens) eine nichttriviale Lösung a_{-r}^*, \dots, a_r^* , die entweder $a_{-r}^* + \dots + a_r^* = 0$ erfüllt, oder die man so normieren kann, daß $a_{-r}^* + \dots + a_r^* = 1$. Auf alle Fälle setzen wir

$$s = \sum_{j=-r}^r a_j^* s_j$$

und da für $j = 0, \dots, n - m + 1$

$$s(t_{m+j}) = \sum_{k=-r}^r a_k^* \underbrace{s_k(t_{j+m})}_{=f_j} = f_j \sum_{k=-r}^r a_k^*,$$

ist s entweder eine Lösung von (9.10) oder ein Lösung des homogenen Problems¹²³ zu (9.10), je nachdem, ob $\sum a_j^*$ den Wert 1 oder 0 hat, aber erfüllt immer auch zusätzlich (9.11). 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 9.11 wäre dann aber $s^{(r+1)} = 0$, also 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 $n - m$! \square

¹²³Also mit rechter Seite 0.

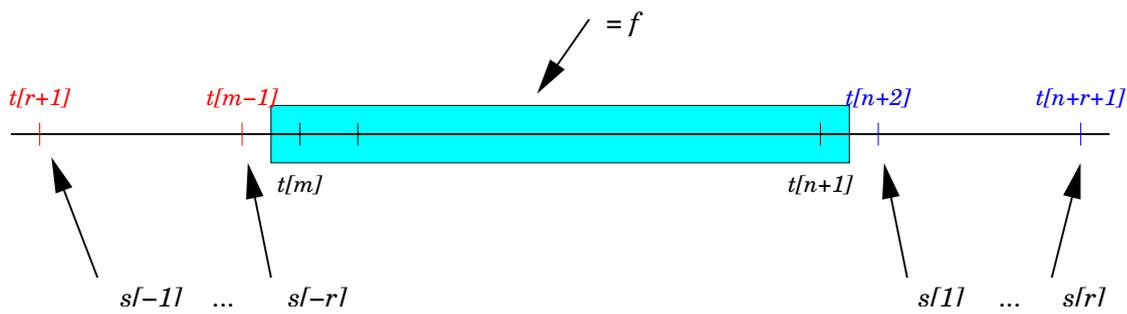


Abbildung 9.5: Das erweiterte Interpolationsproblem aus dem Beweis von Satz 9.9. Auf dem unterlegten Bereich interpolieren alle Splines die vorgegebenen Werte, an den „Zusatzpunkten“ hat genau einer von ihnen den Wert 1, alle andere den Wert 0; nur s_0 verschwindet an *allen* äußeren Punkten.

Für $k \in \mathbb{N}_0$ und $I \subset \mathbb{R}$ definieren wir nun die **Energienorm**¹²⁴

$$|f|_{k,I} = \left(\int_I |f^{(k)}(x)|^2 dx \right)^{1/2}, \quad f \in C^{(k)}(I).$$

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** eines Streifens, der dem Verlauf von f folgt. Daher übrigens auch der Name **Spline**¹²⁵: Ein Spline¹²⁶ ist ein Kurvenlineal, bei dem ein mehr oder weniger beweglicher Streifen mit Hilfe von Gewichten durch gewisse Punkte gezwungen wird, siehe Abb. 9.6. Da der Streifen sich so legt, daß die Biegeenergie¹²⁷ minimiert wird, wird ein **kubischer Spline**¹²⁸ mit den den Randbedingungen aus Satz 9.11¹²⁹ als **natürlicher Spline** bezeichnet weil er das „natürliche“ Objekt „Spline“ simuliert. Dabei entsteht die „Natürlichkeit“ aus den Randbedingungen (9.11) für $m = 3$, also $r = 1$, das heißt aus

$$N''_{m,T} \mathbf{d}(t_m) = N''_{m,T} \mathbf{d}(t_{n+1}) = 0 \tag{9.12}$$

am ersten und letzten Interpolationspunkt, siehe Abb. 9.7. Hierbei kann man (9.12) so interpretieren, 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.

¹²⁴Die allerdings nur eine Halbnorm ist.

¹²⁵Wenn man es genau nimmt, ist also ein Spline gar kein Spline.

¹²⁶Im Schiffsbau als **Straklatte** bezeichnet.

¹²⁷Und die entspricht dem Quadratintegral über die zweite Ableitung, solange die Krümmung im Vergleich zur Ableitung klein ist.

¹²⁸Zweite Ableitung!

¹²⁹Die gerade für die Minimalität sorgen werden.



Abbildung 9.6: 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.

Satz 9.11 (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 $S_f = N_{m,T} \mathbf{d}$ ein Spline, der (9.10) und (9.11) erfüllt. Dann ist*

$$|S_f|_{r+1,I} \leq |f|_{r+1,I} \quad (9.13)$$

Beweis: Wir beginnen mit

$$\begin{aligned} |f - S_f|_{r+1,I}^2 &= \int_I \left(f^{(r+1)}(x) - (S_f)^{(r+1)}(x) \right)^2 dx \\ &= \int_I \left(f^{(r+1)}(x) \right)^2 - 2f^{(r+1)}(x)(S_f)^{(r+1)}(x) + \left((S_f)^{(r+1)}(x) \right)^2 dx \\ &= |f|_{r+1,I}^2 - 2 \int_I \left(f^{(r+1)}(x) - (S_f)^{(r+1)}(x) \right) (S_f)^{(r+1)}(x) dx - |S_f|_{r+1,I}^2. \end{aligned} \quad (9.14)$$

Für $j = m, \dots, n$ verwendet man partielle Integration, um zu zeigen, daß

$$\begin{aligned} &\int_{t_j}^{t_{j+1}} \left(f^{(r+1)}(x) - S_f^{(r+1)}(x) \right) S_f^{(r+1)}(x) dx \\ &= \left(f^{(r)}(x) - S_f^{(r)}(x) \right) S_f^{(r+1)}(x) \Big|_{t_j}^{t_{j+1}} - \int_{t_j}^{t_{j+1}} \left(f^{(r)}(x) - S_f^{(r)}(x) \right) S_f^{(r+2)}(x) dx \\ &= \sum_{l=0}^k (-1)^{r-l} \left(f^{(r-l)}(x) - S_f^{(r-l)}(x) \right) S_f^{(r+l+1)}(x) \Big|_{t_j}^{t_{j+1}} \end{aligned}$$

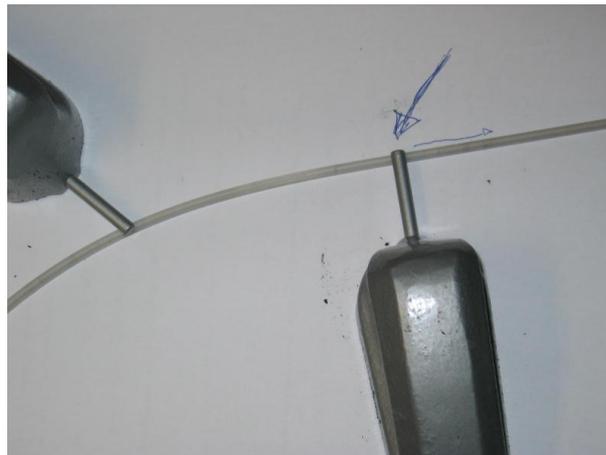


Abbildung 9.7: Die natürliche Randbedingung des kubischen Splines: Rechts vom letzten Knoten und damit Interpolationspunkt werden auf das Lineal keine Biegekräfte mehr ausgeübt und es nimmt den energieminimalen linearen Verlauf an.

$$\begin{aligned}
 & +(-1)^{k+1} \int_{t_j}^{t_{j+1}} \left(f^{(r-k)}(x) - S_f^{(r-k)}(x) \right) \underbrace{S_f^{(r+k+2)}(x)}_{=0 \text{ für } k=r} dx, \quad k = 1, \dots, r \\
 = & \sum_{l=0}^r (-1)^{r-l} \left(f^{(r-l)}(x) - S_f^{(r-l)}(x) \right) S_f^{(r+l+1)}(x) \Big|_{t_j}^{t_{j+1}},
 \end{aligned}$$

Summation über $j = m, \dots, n$ liefert dann, daß¹³⁰

$$\begin{aligned}
 & \int_I \left(f^{(r+1)}(x) - S_f^{(r+1)}(x) \right) S_f^{(r+1)}(x) dx \\
 = & \sum_{l=0}^r (-1)^{r-l} \left(f^{(r-l)}(x) - S_f^{(r-l)}(x) \right) S_f^{(r+l+1)}(x) \Big|_{t_m}^{t_{n+1}} = 0.
 \end{aligned}$$

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 $l = 0, \dots, r - 1$ hingegen wegen (9.11). Einsetzen in (9.14) liefert schließlich

$$|S_f|_{r+1,I}^2 = |f|_{r+1,I}^2 - |f - S_f|_{r+1,I}^2 \leq |f|_{r+1,I}^2.$$

Hierbei gilt Gleichheit genau dann, wenn $f - S_f \in \Pi_r$.

Das zeigt auch die Eindeutigkeit des natürlichen Splineinterpolanten wenn $n \geq m + r$: Ist f irgendeine andere Minimallösung des Interpolationsproblems, dann ist

¹³⁰Die Werte an den inneren Knoten tauchen stets zweimal, aber mit unterschiedlichen Vorzeichen auf.

$f - S_f \in \Pi_r$, muß aber an mindestens $r + 1$ Stellen (den Interpolationspunkten) verschwinden. Also ist $f = S_f$. \square

Übung 9.1 Was sind die kleinsten Zahlen p, q , so daß die Kollokationsmatrix zum natürlichen Splineinterpolant der Ordnung $m = 2r + 1$ eine (p, q) -bandierte Matrix ist. \diamond

Bei de Boor [5] findet man zu diesem Thema die folgende nette Aussage:

Die Extremaleigenschaft des interpolierenden Splines wird häufig für die große praktische Nützlichkeit der Splines verantwortlich gemacht. Dies ist jedoch glatter „Volksbetrug“

9.3 Eigenschaften der B-Splines

Bevor wir uns mit B-Splines befassen, sehen wir uns in Abb. 9.8 erst einmal ein paar davon an. Das erste wichtige Resultat ist eine Rekursionsformel für B-Splines, mit

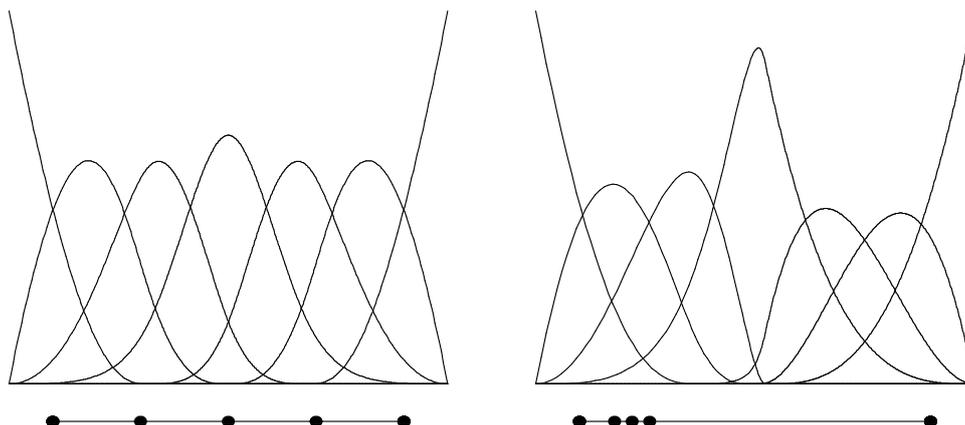


Abbildung 9.8: Eine kleine Kollektion kubischer B-Splines ($m = 3$) mit vierfachen Randknoten. In linken Bild sind die inneren Knoten gleichverteilt, im rechten Bild in der linken Hälfte konzentriert.

deren Hilfe man B-Splines der Ordnung m aus B-Splines der Ordnung $m - 1$ berechnen kann. Diese Formel geht (wieder einmal) auf Carl de Boor zurück.

Satz 9.12 (Rekursionsformel für B-Splines) Sei $T = T_{m,n}$ eine Knotenfolge.

1. Die B-Splines vom Grad 0 haben die Form

$$N_j^0(x|T) = \chi_{[t_j, t_{j+1})}(x) = \begin{cases} 1 & x \in [t_j, t_{j+1}), \\ 0 & x \notin [t_j, t_{j+1}), \end{cases} \quad j = 0, \dots, n + m. \quad (9.15)$$

2. Für $\ell \geq 1$ gilt

$$N_j^\ell(x|T) = u_1(x|I_j^\ell) N_j^{\ell-1}(x|T) + u_0(x|I_{j+1}^\ell) N_{j+1}^{\ell-1}(x|T), \quad (9.16)$$

$$j = 0, \dots, n + m - \ell.$$

Bemerkung 9.13

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 (9.16)

$$N_j^\ell(x|T) = \frac{x - t_j}{t_{j+\ell} - t_j} N_j^{\ell-1}(x|T) + \frac{t_{j+\ell+1} - x}{t_{j+\ell+1} - t_{j+1}} N_{j+1}^{\ell-1}(x|T), \quad (9.17)$$

$$j = 0, \dots, n + m - \ell.$$

3. Die Formel (9.17) 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 ist¹³¹. In diesem Fall läßt man den entsprechenden Term in (9.17) einfach wegfallen.

Lemma 9.14 Die B-Splines sind nichtnegative Funktionen mit kompaktem Träger. Genauer:

$$N_j^m(x|T) > 0, \quad x \in (t_j, t_{j+m+1}), \quad N_j^m(x|T) = 0, \quad x \notin [t_j, t_{j+m+1}]. \quad (9.18)$$

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}, \dots, d_r verwendet werden. Um $N_j^m(x|T) \neq 0$ zu erhalten, muß also $j \in \{r - m, \dots, r\}$ oder eben $r \in \{j, \dots, j + m\}$ 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 $r \in \{j, \dots, j + m\}$, dann sind wegen

$$(t_r, t_{r+1}) \subseteq \left\{ \begin{array}{l} (t_j, t_{j+m-k+1}), \quad j = r - m + k, \dots, r, \\ I_r^{m-k+1}, \end{array} \right. \quad , k = 1, \dots, m,$$

alle baryzentrischen Koordinaten in (9.3) strikt positiv und damit ist auch

$$d_j^1(x) = u_0(x|I_j^m) \underbrace{d_{j-1}^0(x)}_{=d_{j-1}=0} + u_1(x|I_j^m) \underbrace{d_j^0(x)}_{=d_j=1} = u_1(x|I_j^m) > 0$$

sowie

$$d_{j+1}^1(x) = u_0(x|I_{j+1}^m) \underbrace{d_j^0(x)}_{=d_j=1} + u_1(x|I_{j+1}^m) \underbrace{d_{j+1}^0(x)}_{=d_{j+1}=0} = u_0(x|I_{j+1}^m) > 0.$$

¹³¹Genauer: Der Spline wäre, wie wir gleich sehen werden, nur auf der leeren Menge von Null verschieden.

Mit demselben Argument¹³² ergibt sich dann, daß

$$d_k^\ell(x) > 0, \quad k = j, \dots, j + \ell$$

Also ist auch $N_j^m(x|T) = d_r^m(x) > 0$, da $r \in \{j, \dots, j + m\}$. \square

Lemma 9.15 Die B-Splines bilden eine *nichtnegative Teilung der Eins*, d.h.

$$\sum_{j=0}^n N_j^m(\cdot|T) \equiv 1. \quad (9.19)$$

Beweis: Ist $d_0 = \dots = d_n = 1$, also $d_k^0(x) = 1$, $k = 0, \dots, n$, dann ist, nach (9.4) und Induktion über j ,

$$\begin{aligned} d_k^j(x) &= u_0(x|I_k^{m-j+1}) d_{k-1}^{j-1}(x) + u_1(x|I_k^{m-j+1}) d_k^{j-1}(x) \\ &= u_0(x|I_k^{m-j+1}) + u_1(x|I_k^{m-j+1}) = 1, \quad k = j, \dots, m. \end{aligned}$$

\square

Beweis von Satz 9.12: Nach Lemma 9.14 haben die B-Splines $N_j^0(\cdot|T)$, $j = 0, \dots, n + m$, als Träger das Intervall $[t_j, t_{j+1})$, insbesondere sind also die Träger der verschiedenen B-Splines disjunkt und Lemma 9.15 liefert (9.15).

Um $N_j^m(\cdot|T)$ mit Hilfe der B-Splines der Ordnung $m - 1$ ausdrücken zu können, benutzen wir die Zwischenpunkte $d_j^1(x)$, $j = 1, \dots, n$, aus (9.4) und schreiben

$$N_{m,T} \mathbf{d}(x) = \sum_{j=0}^n d_j N_j^m(x|T) = \sum_{j=1}^n d_j^1(x) N_j^{m-1}(x|T). \quad (9.20)$$

Ist insbesondere $d_j = \delta_{jk}$, dann ist, nach (9.3)

$$d_j^1(x) = u_0(x|I_j^m) d_{j-1} + u_1(x|I_j^m) d_j = \begin{cases} u_1(x|I_k^m) & j = k, \\ u_0(x|I_{k+1}^m) & j = k + 1, \\ 0 & \text{sonst.} \end{cases} \quad (9.21)$$

Setzt man (9.21) in (9.20) ein, dann ergibt sich

$$N_j^m(x|T) = u_1(x|I_j^m) N_j^{m-1}(x|T) + u_0(x|I_{j+1}^m) N_{j+1}^{m-1}(x|T). \quad (9.22)$$

\square

¹³²Man ersetzt $d_j^1(x)$ durch $d_j^{\ell+1}(x)$

9.4 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 9.16 Zu $m \in \mathbb{N}$ und einer Knotenfolge $T = T_{m,n}$ ist der **Splineraum** $S_m(T)$ definiert als Menge aller

1. stückweisen Polynome¹³³

$$f \in S_m(T) \quad \Rightarrow \quad f|_{(t_j, t_{j+1})} \in \Pi_m, \quad j = m, \dots, n, \quad (9.23)$$

2. die an einem μ -fachen Knoten t_j , $t_{j-1} < t_j = \dots = t_{j+\mu-1} < t_{j+\mu}$, differenzierbar von der Ordnung $m - \mu$ sind:

$$f \in S_m(T) \quad \Rightarrow \quad f \in C^{m-\mu}(t_{j-1}, t_{j+\mu}). \quad (9.24)$$

Offensichtlich ist $S_m(T)$ ein Vektorraum, das heißt,

$$f, g \in S_m(T) \quad \Rightarrow \quad \alpha f + \beta g \in S_m(T), \quad \alpha, \beta \in \mathbb{R}.$$

Satz 9.17 (Curry–Schoenberg) Für jedes $m \in \mathbb{N}$ und jede Knotenfolge $T = T_{m,n}$ sind die B-Splines $N_j^m(\cdot|T)$, $j = 0, \dots, n$, eine Basis von $S_m(T)$.

Für diesen Satz ist mehr zu beweisen, als man zuerst denken mag: zwar folgt (9.23) 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 $S_m(T)$ tatsächlich $n + 1$ ist. Es gibt einen sehr schönen Beweis von Satz 9.17, 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 Spezialvorlesung¹³⁴. Hier wollen wir den direkten Weg gehen. Dafür müssen wir aber eine zusätzliche Annahme machen, indem wir fordern, daß

$$t_m < t_{m+1} \quad \text{und} \quad t_n < t_{n+1}, \quad (9.25)$$

daß also kein Randknoten als innerer Knoten auftritt¹³⁵. Dies ist insbesondere für $(m + 1)$ -fache Randknoten der Fall.

Eine sehr hilfreiche Aussage, die uns mit ein bißchen Rechenarbeit konfrontieren wird, ist eine Formel für die Ableitung eines B-Splines.

¹³³Nicht vergessen: Eingeschränkt auf die leere Menge hat jede Funktion jede Eigenschaft, deswegen – vorsichtshalber – die offenen Intervalle.

¹³⁴leider ...

¹³⁵Klingt irgendwo plausibel.

Lemma 9.18 Sei $m \in \mathbb{N}$ und $T = T_{m,n}$ eine Knotenfolge. Dann ist, für $x \in \mathbb{R} \setminus T$,

$$\frac{d}{dx} N_j^m(x|T) = \frac{m}{t_{j+m} - t_j} N_j^{m-1}(x|T) - \frac{m}{t_{j+m+1} - t_{j+1}} N_{j+1}^{m-1}(x|T), \quad j = 0, \dots, n. \quad (9.26)$$

Ist t_j oder t_{j+1} ein Knoten der Vielfachheit $m+1$, dann macht (9.26) 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 $U \subset \mathbb{R} \setminus T$ ¹³⁶ ein Polynom ist, können wir immer und fast überall nach Herzenslust differenzieren. Unter Verwendung der Rekursionsformel (9.17) ergibt sich dann

$$\begin{aligned} & \left(\frac{d}{dx} N_j^m(\cdot|T) \right) (x) \\ &= \left(\frac{d}{dx} \left(\frac{\cdot - t_j}{t_{j+m} - t_j} N_j^{m-1}(\cdot|T) + \frac{t_{j+m+1} - \cdot}{t_{j+m+1} - t_{j+1}} N_{j+1}^{m-1}(\cdot|T) \right) \right) (x) \\ &= \frac{1}{t_{j+m} - t_j} N_j^{m-1}(\cdot|T) - \frac{1}{t_{j+m+1} - t_{j+1}} N_{j+1}^{m-1}(\cdot|T) \\ & \quad + \frac{x - t_j}{t_{j+m} - t_j} \frac{d}{dx} \left(N_j^{m-1}(\cdot|T) \right) (x) + \frac{t_{j+m+1} - x}{t_{j+m+1} - t_{j+1}} \frac{d}{dx} \left(N_{j+1}^{m-1}(\cdot|T) \right) (x). \quad (9.27) \end{aligned}$$

Jetzt wird es Zeit für die Induktionshypothese, die zusammen mit (9.17)

$$\begin{aligned} & \frac{x - t_j}{t_{j+m} - t_j} \frac{d}{dx} \left(N_j^{m-1}(\cdot|T) \right) (x) \\ &= \frac{x - t_j}{t_{j+m} - t_j} \left(\frac{m-1}{t_{j+m-1} - t_j} N_j^{m-2}(x|T) - \frac{m-1}{t_{j+m} - t_{j+1}} N_{j+1}^{m-2}(x|T) \right) \\ &= \frac{m-1}{t_{j+m} - t_j} \underbrace{\left(\frac{x - t_j}{t_{j+m-1} - t_j} N_j^{m-2}(x|T) + \frac{t_{j+m} - x}{t_{j+m} - t_{j+1}} N_{j+1}^{m-2}(x|T) \right)}_{= N_j^{m-1}(x|T)} \\ & \quad - \frac{m-1}{t_{j+m} - t_j} \underbrace{\left(\frac{t_{j+m} - x}{t_{j+m} - t_{j+1}} + \frac{x - t_j}{t_{j+m} - t_{j+1}} \right)}_{=(t_{j+m}-t_j)/(t_{j+m}-t_{j+1})} N_{j+1}^{m-2}(x|T) \\ &= \frac{m-1}{t_{j+m} - t_j} N_j^{m-1}(x|T) - \frac{m-1}{t_{j+m} - t_{j+1}} N_{j+1}^{m-2}(x|T) \quad (9.28) \end{aligned}$$

¹³⁶Wir bleiben also immer in einem der „Stückchen“, in die T die reelle Achse zerlegt.

und

$$\begin{aligned}
& \frac{t_{j+m+1} - x}{t_{j+m+1} - t_{j+1}} \frac{d}{dx} \left(N_{j+1}^{m-1}(\cdot|T) \right) (x) \\
&= \frac{t_{j+m+1} - x}{t_{j+m+1} - t_{j+1}} \left(\frac{m-1}{t_{j+m} - t_{j+1}} N_{j+1}^{m-2}(x|T) - \frac{m-1}{t_{j+m+1} - t_{j+2}} N_{j+2}^{m-2}(x|T) \right) \\
&= \frac{m-1}{t_{j+m+1} - t_{j+1}} \underbrace{\left(\frac{x - t_{j+1}}{t_{j+m} - t_{j+1}} + \frac{t_{j+m+1} - x}{t_{j+m} - t_{j+1}} \right)}_{=(t_{j+m+1}-t_{j+1})/(t_{j+m}-t_{j+1})} N_{j+1}^{m-2}(x|T) \\
&\quad - \frac{m-1}{t_{j+m+1} - t_{j+1}} \underbrace{\left(\frac{x - t_{j+1}}{t_{j+m} - t_{j+1}} N_{j+1}^{m-2}(x|T) + \frac{t_{j+m+1} - x}{t_{j+m+1} - t_{j+2}} N_{j+2}^{m-2}(x|T) \right)}_{=N_{j+1}^{m-1}(x|T)} \\
&= \frac{m-1}{t_{j+m} - t_{j+1}} N_{j+1}^{m-2}(x|T) - \frac{m-1}{t_{j+m+1} - t_{j+1}} N_{j+1}^{m-1}(x|T). \tag{9.29}
\end{aligned}$$

liefert. Einsetzen von (9.28) und (9.29) in (9.27) ergibt dann

$$\begin{aligned}
& \left(\frac{d}{dx} N_j^m(\cdot|T) \right) (x) \\
&= \frac{1}{t_{j+m} - t_j} N_j^{m-1}(x|T) - \frac{1}{t_{j+m+1} - t_{j+1}} N_{j+1}^{m-1}(x|T) + \frac{m-1}{t_{j+m} - t_j} N_j^{m-1}(x|T) \\
&\quad - \frac{m-1}{t_{j+m} - t_{j+1}} N_{j+1}^{m-2}(x|T) + \frac{m-1}{t_{j+m} - t_{j+1}} N_{j+1}^{m-2}(x|T) - \frac{m-1}{t_{j+m+1} - t_{j+1}} N_{j+1}^{m-1}(x|T) \\
&= \frac{m}{t_{j+m} - t_j} N_j^{m-1}(x|T) - \frac{m}{t_{j+m+1} - t_{j+1}} N_{j+1}^{m-1}(x|T),
\end{aligned}$$

und somit (9.26). \square

Übung 9.2 Verifizieren Sie (9.26) für $m = 1$. Was bedeutet das geometrisch? \diamond

Damit wissen wir auch, wie die Ableitung einer Splinekurve (bis auf die Werte an den Knoten) aussieht.

Korollar 9.19 Es sein $m \in \mathbb{N}$ und $T = T_{m,n}$ eine Knotenfolge. Dann ist, für alle $x \in \mathbb{R} \setminus T$,

$$\frac{d}{dx} N_{m,T} \mathbf{d}(x) = m \sum_{j=0}^{n+1} \frac{\mathbf{d}_j - \mathbf{d}_{j-1}}{t_{j+m} - t_j} N_j^{m-1}(x|T), \tag{9.30}$$

wobei $\mathbf{d}_{n+1} = \mathbf{d}_{-1} = 0$ ¹³⁷.

¹³⁷Und wieder $N_j^{m-1}(x|T) / (t_{j+m} - t_j) \equiv 0$.

Beweis: Mit Lemma 9.18 ist

$$\begin{aligned} \frac{d}{dx} N_{m,T} \mathbf{d}(x) &= m \sum_{j=0}^n \mathbf{d}_j \left(\frac{N_j^{m-1}(x|T)}{t_{j+m} - t_j} - \frac{N_{j+1}^{m-1}(x|T)}{t_{j+1+m} - t_{j+1}} \right) \\ &= m \sum_{j=0}^{n+1} \frac{\mathbf{d}_j - \mathbf{d}_{j-1}}{t_{j+m} - t_j} N_j^{m-1}(x|T). \end{aligned}$$

□

Proposition 9.20 Sei $m \in \mathbb{N}_0$ und $T = T_{m,n}$. Dann ist $N_j^m(\cdot|T) \in \mathcal{S}_m(T)$, $j = 0, \dots, n$.

Beweis: Induktion über m ¹³⁸. 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 $m - 1 \rightarrow m$, $m \geq 1$, folgt die stückweise Polynomialität direkt aus dem Algorithmus von de Boor, wir brauchen uns also nur noch um die Differenzierbarkeit zu kümmern. An einem $m + 1$ -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 $t \in (t_j, t_{j+m+1})$ ¹³⁹ die Vielfachheit $\mu \leq m$, dann können wir in einem offenen Intervall U mit $U \cap T = \{t\}$ differenzieren und die Ableitungsfunktion ist nach (9.26) 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 9.21 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 $m \rightarrow m + 1$ nehmen wir an, kein Knoten hätte Vielfachheit $m + 1$ und es gäbe einen Knotenvektor $\mathbf{d} = (d_j : j = 0, \dots, n)$, so daß

$$0 = N_{m,T} \mathbf{d} = \sum_{j=0}^n d_j N_j^m(\cdot|T). \quad (9.31)$$

Ableitung beider Seiten liefert für alle $x \in \mathbb{R} \setminus T$, daß

$$0 = \frac{d}{dx} N_{m,T} \mathbf{d} = m \sum_{j=0}^{n+1} \frac{d_j - d_{j-1}}{t_{j+m} - t_j} N_j^{m-1}(\cdot|T).$$

¹³⁸Wie soll man Rekursionsformeln auch sonst ausnutzen?

¹³⁹Also im Inneren des Trägers des B-Splines $N_j^m(\cdot|T)$.

Nach Proposition 9.20 ist die rechte Seite stetig¹⁴⁰ 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 9.3 Ergänzen Sie den Beweis von Lemma 9.21 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 9.22 Für $m \in \mathbb{N}_0$ und eine Knotenfolge $T = T_{m,n}$ ist

$$\dim S_m(T) = n + 1.$$

Beweis: Auf dem, nach (9.25) 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

$$p_{m+1} = \sum_{j=0}^m \frac{\alpha_j}{j!} (x - t_{m+1})^j.$$

Die Differenzierbarkeitsbedingungen sind nun

$$p_m^{(j)}(t_{m+1}) = p_{m+1}^{(j)}(t_{m+1}) = \alpha_j, \quad j = 0, \dots, m - \mu.$$

Also hat der Raum aller stückweisen Polynome mit der geforderten Differenzierbarkeit auf den ersten 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

$$T_\ell = \{t_m, \dots, t_{m+\ell+1}\}, \quad t_m < t_{m+1}, \quad t_{m+\ell} < t_{m+\ell+1},$$

(plus geeignete Randknoten), daß $\dim S_m(T_\ell) = m + \ell + 1$. Für $\ell = n - m$ folgt dann die Behauptung. □

Beweis von Satz 9.17: Der Beweis ist jetzt ein einfaches Spiel mit den schon bewiesenen Bauklötzen: Nach Proposition 9.20 ist

$$\text{span} \{N_j^m(\cdot|T) : j = 0, \dots, n\} \subseteq S_m(T),$$

aber da wegen der linearen Unabhängigkeit der B-Splines (Lemma 9.21) und wegen Lemma 9.22 die Dimension der beiden Vektorräume jeweils $n+1$ ist, müssen sie halt auch übereinstimmen. □

¹⁴⁰Wir haben keine $(m+1)$ -fachen Knoten mehr.

9.5 Knoteneinfügen

Definition 9.23 Eine Knotenfolge $T^* = T_{m,n^*}$ heißt **Verfeinerung** von $T = T_{m,n}$, falls

1. $t_0^* = t_0, \dots, t_m^* = t_m$,
2. $t_{n^*+1}^* = t_{n+1}, \dots, t_{n^*+m+1}^* = t_{n+m+1}$,
3. es gibt $\tau : \{m+1, \dots, n-1\} \rightarrow \{m+1, \dots, n^*-1\}$ monoton steigend, so daß $t_{\tau(j)}^* = t_j, j = m+1, \dots, n-1$.

Formal schreiben wir das als $T \subseteq T^*$.¹⁴¹

Da Polynome unendlich oft differenzierbar sind, insbesondere an „zusätzlichen“ Knoten, ist $S_m(T) \subseteq S_m(T^*)$, wann immer $T \subseteq T^*$. Anders gesagt, für jeden Vektor $\mathbf{d} = (d_0, \dots, d_n)$ gibt es einen Vektor $\mathbf{d}^* = (d_0^*, \dots, d_{n^*}^*)$, so daß

$$N_{m,T} \mathbf{d} = \underbrace{\sum_{j=0}^n d_j N_j^m(\cdot|T)}_{\in S_m(T)} = \underbrace{\sum_{j=0}^{n^*} d_j^* N_j^m(\cdot|T^*)}_{\in S_m(T^*)} = N_{m,T^*} \mathbf{d}^*. \quad (9.32)$$

Die Frage ist natürlich: „Wie bestimmt man \mathbf{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. Um die Situation zu fixieren sei, für ein $j \in \{m, \dots, n-1\}$

$$t_0 \leq \dots \leq t_j \leq t^* \leq t_{j+1} \leq \dots \leq t_{n+m+1},$$

und

$$T^* = \{t_k^* : k = 0, \dots, n+m+2\}, \quad t_k^* = \begin{cases} t_k & k = 0, \dots, j, \\ t^* & k = j+1, \\ t_{k-1} & k = j+2, \dots, n+m+2. \end{cases} \quad (9.33)$$

Der folgende Satz wird normalerweise Boehm zugeschrieben, alternativ spricht man beim Knoteneinfügen auch gerne vom **Oslo-Algorithmus**¹⁴².

Satz 9.24 (Knoteneinfügen) Sei T^* eine Verfeinerung der Knotenfolge $T = T_{m,n}$ gemäß (9.33). Dann ergibt sich das Kontrollpolygon \mathbf{d}^* als

$$\mathbf{d}_k^* = \begin{cases} \mathbf{d}_k, & k = 0, \dots, j-m, \\ u_0 (t^*|I_k^m) \mathbf{d}_{k-1} + u_1 (t^*|I_k^m) \mathbf{d}_k, & k = j-m+1, \dots, j, \\ \mathbf{d}_{k-1}, & k = j+1, \dots, n+1. \end{cases} \quad (9.34)$$

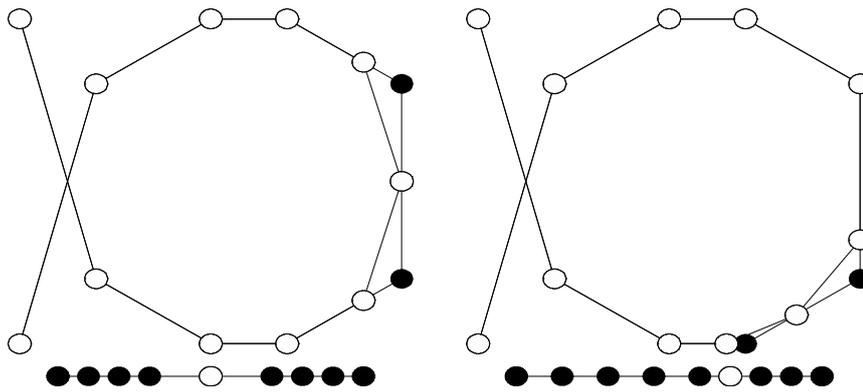


Abbildung 9.9: Zwei Beispiele für Knoteneinfügen bei kubischen Splines, $m = 3$.

Beweis: Übungsaufgaben. □

Übung 9.4 Beweisen Sie die **Leibniz-Formel** für dividierte Differenzen:

$$[x_0, \dots, x_n](fg) = \sum_{j=0}^n [x_0, \dots, x_j] f [x_j, \dots, x_n] g. \quad (9.35)$$

◇

Übung 9.5 Die **abgebrochene Potenz** $(\cdot - y)_+^n$ ist definiert als

$$(x - y)_+^n = \begin{cases} 0, & x \leq y, \\ (x - y)^n, & x > y. \end{cases}$$

Zeigen Sie: Zu einer Knotenfolge $T = T_{m,n}$ ergibt sich der j -te B-Spline als

$$N_j^m(x|T) = [t_j, \dots, t_{j+m+1}] (\cdot - x)_+^m, \quad j = 0, \dots, n,$$

Hinweis: Verwenden Sie die Leibniz-Formel für dividierte Differenzen, (9.35), um nachzuweisen, daß die rechte Seite des obigen Ausdrucks die Rekursionsformel für B-Splines erfüllt. ◇

Übung 9.6 Es sei T^* die Knotenfolge, die entsteht, wenn der Knoten $t_j \leq t^* < t_{j+1}$ in $T = T_{m,n}$ eingefügt wird. Zeigen Sie, daß dann

$$N_k^m(\cdot|T) = \begin{cases} N_k^m(\cdot|T^*), & k = 0, \dots, j - m, \\ u_1(t^*|J_k^m) N_k^m(\cdot|T^*) + u_0(t^*|J_k^m) N_{k+1}^m(\cdot|T^*), & k = j - m + 1, \dots, j, \\ N_{k+1}^m(\cdot|T^*), & k = j + 1, \dots, n, \end{cases}$$

¹⁴¹Die obigen Bedingungen bedeuten eine „ \subset “-Relation, die Vielfachheiten berücksichtigt.

¹⁴²Das Gebiet ist noch jung genug, um Prioritätenkonflikte zu erlauben.

mit $J_k^m = [t_k^*, t_{k+m}^*]$, und folgern Sie daraus die Formel zum Knoteneinfügen.

Hinweis: Verwenden Sie die Darstellung des B-Splines als dividierte Differenz von abgebrochenen Potenzen. \diamond

9.6 Die Schoenberg–Whitney–Bedingung

So, jetzt machen wir uns also an den Beweis des Interpolationssatzes von Schoenberg und Whitney, also Teil 1 von Satz 9.6. Dazu bezeichnen wir mit

$$N_m(\mathcal{X} | T) = [N_k^m(x_j | T) : j, k = 0, \dots, n]$$

die **Kollokationsmatrix** des Interpolationsproblems, das dann die Form

$$N_m(\mathcal{X} | T) \mathbf{d} = \mathbf{f}$$

annimmt und genau dann eindeutig lösbar ist, wenn $\det N_m(\mathcal{X} | T) \neq 0$. Eine Richtung von Satz 9.6 ist jetzt recht einfach.

Proposition 9.25 Sei $m \in \mathbb{N}_0$ und $T = T_{m,n}$ eine Knotenfolge. Ist das Spline-Interpolationsproblem an Stellen x_0, \dots, x_n immer eindeutig lösbar, so ist $t_j < x_j < t_{j+m+1}$, $j = 0, \dots, n$.

Beweis: Nehmen wir an, es gäbe ein j , so daß die **Schoenberg–Whitney–Bedingung** (9.7), also $t_j < x_j < t_{j+m+1}$, verletzt ist. Dann ist entweder $x_j \leq t_j$ oder $x_j \geq t_{j+m+1}$. Im ersten Fall gilt, da der Träger von $N_k^m(\cdot | T)$ das Intervall $[t_k, t_{k+m+1}]$ ist, die Beziehung

$$N_k^m(x_j | T) = 0, \quad k = j, \dots, n,$$

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 $\det N_m(\mathcal{X} | T) = 0$. Im anderen Fall, $x_j \geq t_{j+m+1}$, ist

$$N_k^m(x_j | T) = 0, \quad k = 0, \dots, j,$$

und, da $x_j \leq x_l$, $j \leq l$, gilt natürlich auch

$$N_k^m(x_l | T) = 0, \quad k = 0, \dots, j, \quad l = j, \dots, n.$$

Damit haben die ersten $j + 1$ Spalten von $N_m(\mathcal{X} | T)$ 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. \square

Bemerkung 9.26 *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. Deswegen kann man durchaus einmal zu einem Blick in Kapitel A.4 raten. Für hier soll es aber genug sein mit den Splines.

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 [32]

Nichtlineare Gleichungen | 10

Das Thema dieses Kapitels ist die Lösung beliebiger, auch *nichtlinearer* Gleichungen der Form

$$F(x) = 0, \quad F: \mathbb{R}^n \rightarrow \mathbb{R}^n, \quad n \in \mathbb{N}, \quad (10.1)$$

wobei F natürlich gewissen Voraussetzungen zu genügen hat, und zwar mindestens stetig, meistens sogar differenzierbar sein soll. Daß die rechte Seite von (10.1) den Wert 0 hat, ist keine Einschränkung: Für beliebige rechte Seite $G(x)$ ist ja¹⁴³

$$F(x) = G(x) \quad \Leftrightarrow \quad (F - G)(x) = 0.$$

Beispiel 10.1 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

$$f(x) = 0, \quad f(x) = x^2 - y,$$

und kann über die oft als **Heron-Verfahren** bezeichnete Iteration

$$x^{(0)} = y, \quad x^{(k+1)} = \frac{1}{2} \left(x^{(k)} + \frac{y}{x^{(k)}} \right), \quad k \in \mathbb{N}_0, \quad (10.2)$$

gelöst werden, die sogar schon den Babyloniern bekannt war¹⁴⁴.

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. 10.1 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,

¹⁴³Das setzt natürlich voraus, daß auch G eine „gute“ Funktion im Sinne eines funktionierenden numerischen Verfahrens ist.

¹⁴⁴Siehe z.B. [10]

siehe [31, S. 125–127]. Laut [29] sind Keilschrifttexte wesentlich stärker algebraisch als geometrisch orientiert – ganz im Gegensatz zu den Rechenmeistern der Renaissance, siehe z.B. [21].

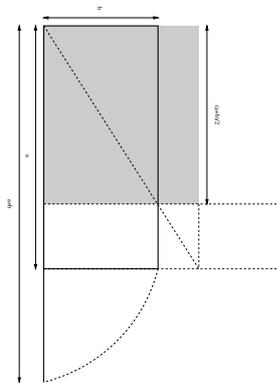


Abbildung 10.1: Ein Iterationsschritt (10.2) des Heron-Verfahrens geometrisch: Konstruiere zu vorgegebenen Seitenlängen a und b das flächengleiche Rechteck mit den Seitenlängen $\frac{a+b}{2}$ und $\frac{2ab}{a+b}$.

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, nichts anderes bedeutet Wurzelziehen, denn die Seitenlänge des Quadrats ist ja dann die gesuchte Wurzel. 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

$$x^* = \alpha x + (1 - \alpha) \frac{y}{x}, \quad \alpha \in (0, 1), \quad (10.3)$$

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 und naheliegendes, 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 (10.3) für jedes α auch dieses x zurück – das ist eben die Fixpunkteigenschaft.

Übung 10.1 Zeigen Sie, daß das Heron-Verfahren immer gegen die Quadratwurzel von y konvergiert. Hinweis: Bestimmen Sie die Fixpunkte der Iteration. \diamond

10.1 Terminologie

Definition 10.2 Ein iteratives Verfahren $x^{(k)} \in \mathbb{R}^n$, $k \in \mathbb{N}_0$, mit $\lim_{k \rightarrow \infty} x^{(k)} = x$ hat die **Konvergenzordnung** $p \geq 1$, bezüglich des Fehlermaßes¹⁴⁵ $\varepsilon_k \geq 0$, $k \in \mathbb{N}$, $\lim_{k \rightarrow \infty} \varepsilon_k = 0$ wenn es eine Konstante $C > 0$ gibt, so daß

$$\limsup_{k \rightarrow \infty} \frac{\varepsilon_{k+1}}{\varepsilon_k^p} = C. \quad (10.4)$$

Ist $p = 1$, so muß außerdem $C < 1$ sein.

Anschaulich heißt das, daß ε_k wie $C \rho^{p^k}$ für ein $0 < \rho < 1$ gegen Null konvergiert.

Definition 10.3 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.

10.2 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, daß¹⁴⁶

$$f(x^{(0)}) < 0, \quad f(y^{(0)}) > 0,$$

dann setzen wir, für $k \in \mathbb{N}_0$,

$$\begin{aligned} z &= \frac{x^{(k)} + y^{(k)}}{2} \\ x^{(k+1)} &= \begin{cases} x^{(k)} & f(z) > 0, \\ z & f(z) < 0, \end{cases} \\ y^{(k+1)} &= \begin{cases} z & f(z) > 0, \\ y^{(k)} & f(z) < 0. \end{cases} \end{aligned} \quad (10.5)$$

Ist zufällig $f(z) = 0$, dann haben wir ohnehin die Nullstelle gefunden.

Satz 10.4 Ist $f \in C[a, b]$ und sind $x^{(0)}, y^{(0)} \in [a, b]$ so daß

$$f(x^{(0)}) < 0, \quad f(y^{(0)}) > 0,$$

dann konvergieren die Folgen $x^{(k)}, y^{(k)}$, $k \in \mathbb{N}_0$, definiert durch (10.5) von linearer Ordnung gegen eine Nullstelle x von f .

¹⁴⁵Normalerweise definiert durch Abschätzungen der Form $\varepsilon_k \geq \|x^{(k)} - x\|$.

¹⁴⁶Wäre f an einer der beiden Stellen $= 0$, dann können wir uns das Verfahren sparen!

```
##  
## Bisekt.m  
## Bisektionsverfahren  
## Daten:  
## f  Funktion  
## 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 10.1 Bisekt.m: Das Bisektionsverfahren.

Bemerkung 10.5 (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.
2. Die Konvergenzaussage gilt¹⁴⁷ nicht nur für Intervalle, sondern für beliebige konvexe metrische Räume, also insbesondere für Funktionen $f \in C(\mathbb{R}^n)$. Damit ist es auch besonders flexibel, weil man nicht viel von f wissen muss, Hauptsache, f kann ausgewertet werden.
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 10.4: Wir können ohne Einschränkung annehmen, daß $x^{(0)} < y^{(0)}$ ist¹⁴⁸, und halten fest, daß das Bisektionsverfahren 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 Fall¹⁴⁹ immer eine Nullstelle $t_k \in (a_k, b_k)$, und da

$$t_k - a_k \leq b_k - a_k \leq \frac{b - a}{2^k} \quad \text{sowie} \quad b_k - t_k \leq b_k - a_k \leq \frac{b - a}{2^k}$$

¹⁴⁷Ohne echten Mehraufwand im Beweis.

¹⁴⁸Andernfalls funktioniert der Beweis ganz genauso!

¹⁴⁹Und nur der ist interessant, denn im zweiten Fall haben wir ja schon eine Nullstelle gefunden.

ist, konvergieren sogar beide Folgen gegen eine¹⁵⁰ 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. □

10.3 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. Jhd.), der sie aus im Arabischen als **Elchataym** kennenlernte und lateinisch als

regula duarum falsarum positionum

bezeichnete. Bei ihm ist sie **die** Methode, mit der alle mathematischen Probleme gelöst werden können, siehe [39, Kap. 13, S. 447]. Zur Erklärung des Namens sagt laut [10, S. 245] der Rechenmeister Peter Bienewitz (1527)

Vnd heisst nit darum falsi dass¹⁵¹ sie falsch und unrecht wehr, sunder, dass sie auss zweyen falschen vnd vnwahrhaftigen zalen und zweyen lügen die wahrhaftige vnd beehrte 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 [32]:

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 gewesen¹⁵².

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

$$f(x^{(k)}) \cdot f(y^{(k)}) < 0,$$

dann unterteilt man nicht am Mittelpunkt des Intervalls $[x^{(k)}, y^{(k)}]$, sondern an der Nullstelle z der Geraden ℓ mit

$$\ell(x^{(k)}) = f(x^{(k)}), \quad \ell(y^{(k)}) = f(y^{(k)}).$$

¹⁵⁰Vorsicht: 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.

¹⁵¹Was zeigt, wie progressiv die Rechtschreibreform 1999 wirklich war.

¹⁵²Für die, die es nicht gleich begriffen haben: Zu lösen ist die Gleichung $\frac{5}{2}x = 30$ und tatsächlich kommt – nach Adam Riese eben – auch 12 heraus.

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

$$z = (1-\alpha)x^{(k)} + \alpha y^{(k)}, \quad \alpha = \frac{f\left(x^{(k)}\right)}{f\left(x^{(k)}\right) - f\left(y^{(k)}\right)}, \quad (10.6)$$

und der Iterationsschritt ist wie beim Bisektionsverfahren

$$x^{(k+1)} = \begin{cases} x^{(k)} & f\left(x^{(k)}\right) f(z) > 0, \\ z & f\left(x^{(k)}\right) f(z) < 0, \end{cases} \quad y^{(k+1)} = \begin{cases} y^{(k)} & f\left(y^{(k)}\right) f(z) > 0, \\ z & f\left(y^{(k)}\right) f(z) < 0. \end{cases} \quad (10.7)$$

Übrigens wurde die Regula Falsi auch dazu benutzt, Gleichungen in *zwei* Unbekan-

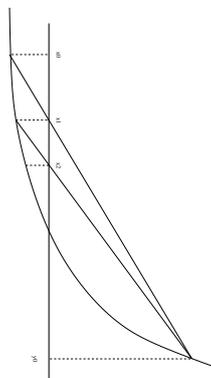


Abbildung 10.2: Geometrische Interpretation der Regula Falsi.

nten zu lösen. Hier ein Beispiel aus [32]:

*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?*¹⁵³

Übung 10.2 Zeigen Sie, daß die Regula Falsi in einem reellen Intervall konvergiert und Konvergenzordnung 1 hat. \diamond

¹⁵³Also $x + \frac{1}{3}y = \frac{1}{4}x + y = 15$ 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.

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

$$x^{(k+1)} = x^{(k)} - \frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})} f(x^{(k)}), \quad (10.8)$$

ihre Herleitung ist wie die von (10.6). Das Schöne am Sekantenverfahren ist nun, daß

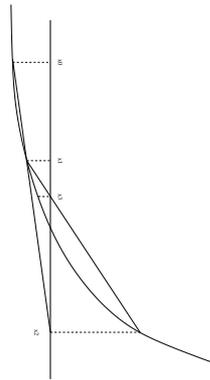


Abbildung 10.3: Geometrische Interpretation des Sekantenverfahrens.

es lokal konvergent mit einer höheren Konvergenzordnung als 1 ist.

Satz 10.6 *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 $p = \frac{1}{2}(1 + \sqrt{5})$.*

Die Zahl $p = \frac{1}{2}(1 + \sqrt{5})$ heißt **Goldener Schnitt** und ist der Grenzwert zweier aufeinanderfolgender **Fibonacci-Zahlen**. Der goldene Schnitt spielt als Proportion eine (mehr oder weniger) zentrale und oft überschätzte Rolle in Kunst und Architektur, aber auch in der Natur.

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

$$\lim_{\varepsilon \rightarrow 0} M_\varepsilon = \left| \frac{f''(x)}{2f'(x)} \right| < \infty.$$

Also gibt es ein ε_0 , so daß für alle $\varepsilon < \varepsilon_0$

$$B_\varepsilon(x) \subseteq U_x \quad \text{und} \quad \varepsilon M_\varepsilon =: \gamma < 1 \quad \text{und} \quad 0 \notin f'(B_\varepsilon(x)).$$

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 $y \in B_\varepsilon(x)$

$$f(y) = \underbrace{f(x)}_{=0} + (y-x)f'(x) + \frac{(y-x)^2}{2}f''(\xi), \quad \xi \in [x, y] \subset B_\varepsilon(x),$$

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 (10.8). Dann gilt für $k \in \mathbb{N}$ unter Verwendung der dividierten Differenzen, siehe Definition 8.18, solange¹⁵⁴ $x^{(k-1)}, x^{(k)} \neq x$

$$\begin{aligned} x^{(k+1)} - x &= x^{(k)} - \underbrace{\frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})}}_{=1/[x^{(k-1)}, x^{(k)}]f} f(x^{(k)}) - x = x^{(k)} - x - \frac{f(x^{(k)})}{[x^{(k-1)}, x^{(k)}]f} \\ &= (x^{(k)} - x) \left(1 - \frac{f(x^{(k)}) - f(x)}{x^{(k)} - x} \frac{1}{[x^{(k-1)}, x^{(k)}]f} \right) \\ &= (x^{(k)} - x) \left(1 - \frac{[x^{(k)}, x]f}{[x^{(k-1)}, x^{(k)}]f} \right) \\ &= \frac{(x^{(k)} - x)(x^{(k-1)} - x)}{[x^{(k-1)}, x^{(k)}]f} \underbrace{\frac{[x^{(k-1)}, x^{(k)}]f - [x^{(k)}, x]f}{x^{(k-1)} - x}}_{=[x^{(k-1)}, x^{(k)}, x]f} \\ &= (x^{(k)} - x)(x^{(k-1)} - x) \frac{[x^{(k-1)}, x^{(k)}, x]f}{[x^{(k-1)}, x^{(k)}]f} \end{aligned}$$

¹⁵⁴Ansonsten sind wir sowieso fertig und haben die Nullstelle gefunden!

Sind nun $x^{(k-1)}, x^{(k)} \in B_\varepsilon(x)$, dann ist

$$\left[x^{(k-1)}, x^{(k)} \right] f = f'(\xi_1), \quad \left[x^{(k-1)}, x^{(k)}, x \right] f = \frac{1}{2} f''(\xi_2), \quad \xi_1, \xi_2 \in B_\varepsilon(x),$$

und damit ist

$$\left| x^{(k+1)} - x \right| = \underbrace{\left| x^{(k)} - x \right|}_{\leq \varepsilon} \underbrace{\left| x^{(k-1)} - x \right|}_{\leq \varepsilon} \underbrace{\left| \frac{f''(\xi_2)}{2f'(\xi_1)} \right|}_{\leq M_\varepsilon} \leq \varepsilon^2 M_\varepsilon < \varepsilon, \quad (10.9)$$

also ist auch $x^{(k+1)} \in B_\varepsilon(x)$. Außerdem ist

$$\left| x^{(k+1)} - x \right| \leq \varepsilon M_\varepsilon \left| x^{(k)} - x \right| \leq (\varepsilon M_\varepsilon)^2 \left| x^{(k-1)} - x \right| \leq \dots \leq \gamma^{k+1} \underbrace{\left| x^{(0)} - x \right|}_{\leq \varepsilon},$$

und deswegen konvergiert die Folge $x^{(k)} \rightarrow x$ für $k \rightarrow \infty$.

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 (10.9), 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 $E = \max \{ E_0, E_1^{1/p} \}$ ist $E_k \leq E^{p^k}$ für $k = 0, 1$, und generell folgt für $k \geq 1$ unter Verwendung der Identität¹⁵⁵ $p^2 = p + 1$, daß

$$E_{k+1} \leq E_k E_{k-1} \leq E^{p^k + p^{k-1}} = E^{p^{k-1}(p+1)} = E^{p^{k-1}p^2} = E^{p^{k+1}}.$$

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 [32, 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

¹⁵⁵Sie definiert den goldenen Schnitt.

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.

10.4 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. Die Iterationsvorschrift lautet jetzt also

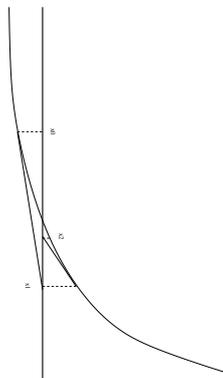


Abbildung 10.4: Geometrische Interpretation des Newton–Verfahrens.

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}, \quad k \in \mathbb{N}_0, \quad x^{(0)} \in \mathbb{R}; \quad (10.10)$$

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 10.7 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 10.6. Über die verallgemeinerte Definition¹⁵⁶ der dividierten Differenzen mit mehrfachen Knoten,

$$\underbrace{[x, \dots, x]}_{n+1} f := \frac{f^{(n)}(x)}{n!},$$

ist nun ganz analog

$$\begin{aligned} x^{(k+1)} - x &= x^{(k)} - x - \frac{f(x^{(k)})}{f'(x^{(k)})} = (x^{(k)} - x) \left(1 - \frac{f(x^{(k)}) - f(x)}{(x^{(k)} - x) f'(x^{(k)})} \right) \\ &= (x^{(k)} - x) \left(1 - \frac{[x^{(k)}, x] f}{[x^{(k)}, x^{(k)}] f} \right) \\ &= \frac{(x^{(k)} - x)^2}{[x^{(k)}, x^{(k)}] f} \frac{[x^{(k)}, x^{(k)}] f - [x^{(k)}, x] f}{(x^{(k)} - x)} = (x^{(k)} - x)^2 \frac{[x^{(k)}, x^{(k)}, x] f}{[x^{(k)}, x^{(k)}] f} \\ &= (x^{(k)} - x)^2 \frac{f''(\xi_2)}{2f'(\xi_1)}. \end{aligned}$$

Der übrige Konvergenzbeweis läuft wieder wie bei Satz 10.6, die Konvergenzordnung ergibt sich aus

$$\lim_{k \rightarrow \infty} \frac{|x^{(k+1)} - x|}{|x^{(k)} - x|^2} = \left| \frac{f''(x)}{2f'(x)} \right|.$$

□

Das Newton–Verfahren ist eine Iteration der Form

$$x^{(k+1)} = \Phi(x^{(k)}), \quad k \in \mathbb{N}_0, \quad (10.11)$$

und das Auffinden einer Nullstelle von f bedeutet nichts anderes, als einen *Fixpunkt* von Φ zu finden. Das heißt, falls $f'(x) \neq 0$, dann ist

$$f(x) = 0 \quad \Leftrightarrow \quad x = \Phi(x) := x - \frac{f(x)}{f'(x)}.$$

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 formulieren¹⁵⁷.

¹⁵⁶Diese Erweiterung ist konsistent! Siehe Korollar 8.23.

¹⁵⁷Den Banachschen Fixpunktsatz kann man gar nicht oft genug beweisen und eine schöne Wiederholung ist's allemal.

Satz 10.8 (Banachscher Fixpunktsatz, metrisch) Sei X ein vollständiger metrischer Raum und $\Phi : X \rightarrow X$ eine **Kontraktion**, das heißt,

$$d(\Phi(x), \Phi(y)) \leq \gamma d(x, y), \quad 0 < \gamma < 1, \quad x, y \in X.$$

Dann besitzt Φ einen eindeutig bestimmten **Fixpunkt** $x^* = \Phi(x^*)$ und für jeden Startwert $x^{(0)} \in X$ konvergiert die Folge $x^{(k+1)} = \Phi(x^{(k)})$, $k \in \mathbb{N}_0$, gegen x^* und es ist

$$d(x^*, x^{(k)}) \leq \frac{\gamma^k}{1-\gamma} d(x^{(1)}, x^{(0)}). \quad (10.12)$$

Beweis: Beginnen wir mit der Eindeutigkeit. Für zwei Fixpunkte $x^*, y^* \in X$ ist

$$d(x^*, y^*) = d(\Phi(x^*), \Phi(y^*)) \leq \gamma d(x^*, y^*) \quad \Rightarrow \quad x^* = y^*.$$

Für die Existenz des Fixpunkts bemerken wir zuerst, daß für $k \in \mathbb{N}_0$,

$$d(x^{(k+1)}, x^{(k)}) = d(\Phi(x^{(k)}), \Phi(x^{(k-1)})) \leq \gamma d(x^{(k)}, x^{(k-1)}) \leq \dots \leq \gamma^k d(x^{(1)}, x^{(0)}),$$

und somit, für $m > 0$,

$$d(x^{(k+m)}, x^{(k)}) \leq \sum_{j=0}^{m-1} d(x^{(k+j+1)}, x^{(k+j)}) \leq \gamma^k d(x^{(1)}, x^{(0)}) \sum_{j=0}^{m-1} \gamma^j \leq \frac{\gamma^k}{1-\gamma} d(x^{(1)}, x^{(0)}),$$

also ist $x^{(k)}$, $k \in \mathbb{N}_0$, eine **Cauchyfolge**, die wegen der Vollständigkeit¹⁵⁸ von X gegen einen Grenzwert x^* konvergiert, für den offensichtlich (10.12) gilt. Weil für beliebiges $k \in \mathbb{N}_0$

$$\begin{aligned} d(\Phi(x^*), x^*) &\leq d(\Phi(x^*), \Phi(x^{(k)})) + d(x^{(k+1)}, x^{(k)}) + d(x^*, x^{(k)}) \\ &\leq (1+\gamma) d(x^*, x^{(k)}) + \gamma^k d(x^{(1)}, x^{(0)}) \end{aligned}$$

und weil die rechte Seite für $k \rightarrow \infty$ gegen 0 konvergiert, ist x^* der gesuchte Fixpunkt. \square

Nun können wir ein einfaches Konvergenzkriterium für Fixpunktverfahren mit Funktionen $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ geben. Dafür erinnern wir uns an die Definition des **Spektralradius** einer Matrix $A \in \mathbb{R}^{n \times n}$ als

$$\rho(A) = \max\{|\lambda| : \ker_{\mathbb{C}^n}(A - \lambda I) \neq \{0\}\} = \liminf_{k \rightarrow \infty} \|A^k\|^{1/k}. \quad (10.13)$$

Außerdem bezeichnen wir die **Jacobi-Matrix**¹⁵⁹ einer differenzierbaren Funktion $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ mit $J[\Phi]$:

$$J[\Phi] = \left[\frac{\partial \Phi_j}{\partial x_k} : j, k = 1, \dots, n \right].$$

¹⁵⁸Wie man sieht wird in diesem Bereich wirklich jede Voraussetzung gebraucht!

¹⁵⁹Das ist nun schon die zweite Jacobi-Matrix, die hier auftaucht, aber, Kontext sei Dank, Verwechslungen sollten eigentlich ausgeschlossen sein.

Proposition 10.9 Ist $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ 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 $\rho(J[\Phi](x)) < 1$, dann ist das Iterationsverfahren $x^{(k+1)} = \Phi(x^{(k)})$, $k \in \mathbb{N}_0$, lokal konvergent gegen x .

Beweis: Ist $\rho(J[\Phi](x)) < 1$, dann gibt es¹⁶⁰ eine Vektornorm $\|\cdot\|$, so daß $\|J[\Phi](x)\| < 1$ und wegen der stetigen Differenzierbarkeit von Φ und der Stetigkeit der Norm gibt es für jedes $\|J[\Phi](x)\| < \gamma < 1$ ein $\varepsilon > 0$, so daß

$$\sup_{y \in B_\varepsilon(x)} \|J[\Phi](y)\| < \gamma.$$

Nach dem Hauptsatz der Differential- und Integralrechnung ist für $y, y' \in B_\varepsilon(x)$

$$\begin{aligned} \Phi(y) - \Phi(y') &= \int_0^1 \frac{d}{dt} \Phi(y' + t(y - y')) \, dt = \int_0^1 \frac{d}{dt} \Phi(ty + (1-t)y') \, dt \\ &= \int_0^1 D_{y-y'} \Phi(ty + (1-t)y') \, dt \\ &= \int_0^1 (J[\Phi](ty + (1-t)y')) (y - y') \, dt, \end{aligned}$$

also

$$\|\Phi(y) - \Phi(y')\| \leq \max_{\xi \in B_\varepsilon(x)} \|J[\Phi](\xi)\| \|y - y'\| = \gamma \|y - y'\|,$$

und damit ist Φ auf $B_\varepsilon(x)$ eine Kontraktion, die nach Satz 10.8 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

$$\|x^{(k+1)} - x\| = \|\Phi(x^{(k)}) - \Phi(x)\| \leq \gamma \|x^{(k)} - x\| \leq \gamma \varepsilon < \varepsilon.$$

□

Übung 10.3 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. ◇

Bemerkung 10.10 (Zu Proposition 10.9)

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.

¹⁶⁰Siehe Übungsaufgabe 10.3.

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 $(-\frac{1}{2}, \frac{1}{2})$ kleiner als 1 und damit ist auch $|\Phi(x) - \Phi(y)| < |x - y|$ für $x, y \in (-\frac{1}{2}, \frac{1}{2})$, aber mit $x^{(0)} = 0$ erhalten wir die divergente Iterationsfolge

$$x^{(1)} = 1, \quad x^{(2)} = 2, \quad x^{(3)} = 5, \quad \dots$$

Die Funktion $\Phi(x) = x^2 - 1$ hat hingegen zwei Fixpunkte, nämlich $\frac{1}{2}(1 \pm \sqrt{5})$, 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 $x = \frac{1}{2}(1 \pm \sqrt{5}) + y =: x_1 + y, y > 0$, dann ist

$$\Phi(x) = \underbrace{-1 + x_1^2}_{=x_1} + 2x_1y + y^2 = x_1 + 2x_1y + y^2 > x_1 + 2y,$$

und damit divergiert die Iteration 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

$$x = \Phi(x) \quad \text{und} \quad D^k \Phi(x) = 0, \quad k = 1, \dots, p-1,$$

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

$$\begin{aligned} x^{(k+1)} - x &= \Phi(x^{(k)}) - \Phi(x) \\ &= \underbrace{\sum_{j=1}^{p-1} \sum_{|\alpha|=j} \frac{1}{\alpha!} \frac{\partial^j \Phi}{\partial x^\alpha}(x) (x^{(k)} - x)^\alpha}_{=0} + \sum_{|\alpha|=p} \frac{1}{\alpha!} \frac{\partial^p \Phi}{\partial x^\alpha}(x) (x^{(k)} - x)^\alpha + O\left(\|x^{(k)} - x\|^{p+1}\right), \end{aligned}$$

und wenn $x^{(k)} \rightarrow x$ für $k \rightarrow \infty$, dann ist

$$\limsup_{k \rightarrow \infty} \frac{\|x^{(k+1)} - x\|_\infty}{\|x^{(k)} - x\|_\infty^p} \leq \max_{|\alpha|=p} \left| \frac{\partial^p \Phi}{\partial x^\alpha}(x) \right|,$$

was auf Konvergenzordnung $\geq p$ schließen lässt¹⁶¹

Mit dieser Theorie können wir jetzt auch das Newton-Verfahren für ein **nichtlineares Gleichungssystem**, also unser $F(x) = 0$ aus (10.1), in Angriff nehmen. Ist nämlich $F: \mathbb{R}^n \rightarrow \mathbb{R}^n$ differenzierbar, so können wir die Iteration

$$x^{(k+1)} = x^{(k)} - J^{-1}[F](x^{(k)}) F(x^{(k)}), \quad k \in \mathbb{N}_0, \quad x^{(0)} \in \mathbb{R}^n, \quad (10.14)$$

verwenden. Und siehe da – dieses Verfahren konvergiert!

¹⁶¹Das 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 10.11 Ist $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ zweimal stetig differenzierbar und ist $x \in \mathbb{R}^n$ eine einfache Nullstelle von F , das heißt,

$$F(x) = 0, \quad \text{und} \quad \det J[F](x) \neq 0, \quad (10.15)$$

dann ist das Newton-Verfahren (10.14) lokal konvergent gegen x .

Beweis: Nach Proposition 10.9 müssen wir „nur“ zeigen, daß die Iterationsfunktion

$$\Phi(x) := x - J^{-1}[F](x) F(x), \quad x \in \mathbb{R}^n,$$

die Bedingung $\rho(J[\Phi](x)) < 1$ erfüllt. Aus der zweimaligen Differenzierbarkeit von F werden wir sogar mehr bekommen, nämlich

$$J[\Phi](x) = 0. \quad (10.16)$$

Da F zweimal stetig differenzierbar ist, ist für $y \in \mathbb{R}^n$

$$F(x+y) = \underbrace{F(x)}_{=0} + J[F](x) y + \underbrace{\left[\frac{1}{2} y^T \left[\frac{\partial^2 F_\ell}{\partial x_j \partial x_k}(\xi_{jk}) \right] y : \ell = 1, \dots, n \right]}_{=: G(x,y)},$$

wobei $\xi_{j,k} \in [x, x+y]$, $j, k = 1, \dots, n$, und

$$\lim_{y \rightarrow 0} \left[\frac{\partial^2 F_\ell}{\partial x_j \partial x_k}(\xi_{jk}) \right] = \left[\frac{\partial^2 F_\ell}{\partial x_j \partial x_k}(x) \right] =: H[F_\ell](x), \quad \ell = 1, \dots, n,$$

weil F zweimal stetig differenzierbar ist¹⁶². Daraus können wir nun folgern, daß es eine Konstante $C > 0$ gibt, so daß

$$\|G(x, y)\|_\infty \leq C \max_{\ell=1, \dots, n} \left\| \left[\frac{\partial^2 F_\ell}{\partial x_j \partial x_k}(x) \right] \right\|_\infty \|y\|_\infty^2. \quad (10.17)$$

Damit erhalten wir, daß

$$\begin{aligned} \Phi(x+y) - \Phi(x) &= (x+y) - J^{-1}[F](x+y) F(x+y) - x + J^{-1}[F](x) F(x) \\ &= y - J^{-1}[F](x) (F(x+y) - F(x)) + \left(J^{-1}[F](x) - J^{-1}[F](x+y) \right) F(x+y) \\ &= y - J^{-1}[F](x) (J[F](x) y + G(x, y)) \\ &\quad + \left(J^{-1}[F](x) - J^{-1}[F](x+y) \right) \underbrace{(F(x) + J[F](\xi) y)}_{=0} \\ &= -J^{-1}[F](x) G(x, y) + \left(J^{-1}[F](x) - J^{-1}[F](x+y) \right) J[F](\xi) y, \end{aligned}$$

¹⁶²Die $n \times n$ -Matrix $H[f] = \left[\frac{\partial^2 f}{\partial x_j \partial x_k} : j, k = 1, \dots, n \right]$ 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

$$\left\| J^{-1}[F](x) - J^{-1}[F](x + y) \right\|_{\infty} \leq D \|y\|_{\infty},$$

also ist, für hinreichend kleine y

$$\left\| \left(J^{-1}[F](x) - J^{-1}[F](x + y) \right) J[F](\xi) y \right\|_{\infty} \leq (D + 2\pi) \|J[F](x)\|_{\infty} \|y\|_{\infty}^2. \quad (10.18)$$

Aus (10.17) und (10.18) folgt nun, daß es eine¹⁶³ Konstante $E > 0$ gibt, so daß

$$\| \Phi(x + y) - \Phi(x) \|_{\infty} \leq E \|y\|_{\infty}^2 \quad \Rightarrow \quad \frac{\| \Phi(x + y) - \Phi(x) \|_{\infty}}{\|y\|_{\infty}} \leq E \|y\|_{\infty}.$$

Dies bedeutet aber nichts anderes als daß Φ an x differenzierbar ist und $J[\Phi](x) = 0$. \square

Korollar 10.12 *Das Newton–Verfahren (10.14) konvergiert für zweimal stetig differenzierbare Funktionen F lokal von quadratischer Ordnung gegen eine einfache Nullstelle x von F .*

Bemerkung 10.13 (Newtonverfahren)

1. Die große Schwäche des Newtonverfahrens und letztlich aller Fixpunktiterationen ist die nur 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 reagieren sehr empfindlich auf eine **doppelte Nullstellen**, bei der zumindest die Konvergenzordnung auf linear sinkt. Das an sich wäre ja noch zu verkraften, aber die fast waagerechten Tangenten in der Nähe der Lösung führen zu sehr kleinen Divisoren und zu einer großen Empfindlichkeit gegen Rundungsfehler.
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 Horner Schema einfach mitberechnet werden können, bei komplizierteren Funktionen hingegen muß man numerisch differenzieren, was den einzelnen Iterationsschritt sehr aufwendig machen kann.
4. Beim **Broyden–Verfahren** [6] wird dieses Problem umgangen, indem man aus einer näherungsweise Jacobi–Matrix B_k an $x^{(k)}$ über die Iteration

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - \lambda_k B_k^{-1} F(x^{(k)}) \\ B_{k+1} &= B_k + \frac{1}{\|x^{(k+1)} - x^{(k)}\|_2^2} (\Delta F - B_k \Delta x) (\Delta x)^T, \end{aligned}$$

¹⁶³letzte ...

eine neue näherungsweise Jacobi–Matrix B_{k+1} bestimmt, wobei $\Delta F = F(x^{(k+1)}) - F(x^{(k)})$ 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 [41, S. 246–249] und vor allem auch in [26].

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 Konvergenzordnung¹⁶⁴ ≥ 1 gegen x , dann konvergiert die Folge

$$y^{(k)} := x^{(k)} - \frac{(\Delta x^{(k)})^2}{\Delta^2 x^{(k)}} = x^{(k)} - \frac{(x^{(k+1)} - x^{(k)})^2}{x^{(k+2)} - 2x^{(k+1)} + x^{(k)}}$$

schneller gegen x , es gilt nämlich

$$\lim_{k \rightarrow \infty} \frac{y^{(k)} - x}{x^{(k)} - x} = 0.$$

Details finden sich beispielsweise in [41, S. 274–279].

Ein paar ergänzende Anmerkung, zum einen ein „realistisches“ Beispiel für ein nicht-lineares System, zum anderen die Anwendung des Newtonverfahrens auf Polynome, finden sich im Anhang unter A.5 und A.6.

¹⁶⁴Achtung: Das bedeutet bereits *exponentielle* Konvergenz.

Das Erlernen der deutschen Sprache ist für das Gelingen der Integration von zentraler Bedeutung und wird daher gefördert.

Teilhabe- und Integrationsgesetz
NRW, §2 (3)

Numerische Integration und orthogonale Polynome

11

In vielen praktischen Anwendungen steht man vor der Aufgabe, ein **Integral** zu berechnen. Leider lassen sich nur die wenigsten Funktionen, beispielsweise Polynome, wirklich *explizit* integrieren, für allgemeinere Funktionen muß man *numerische* Methoden finden.

Das Ziel in diesem Abschnitt soll es sein, (eigentliche) Integrale der Form

$$I(f) = \int_a^b f(x) w(x) dx, \quad w(x) \geq 0, x \in [a, b], \quad (11.1)$$

bezüglich einer stetigen¹⁶⁵ und nichtnegativen **Gewichtsfunktion**¹⁶⁶ w näherungsweise zu berechnen. Das "richtige" Kriterium ist hier nicht die Positivität oder Nichtnegativität der Gewichtsfunktion sondern die **strikte Quadratpositivität** für das **Funktional**¹⁶⁷ I , die fordert, daß

$$I(f^2) > 0, \quad f \neq 0. \quad (11.2)$$

Das und nichts anderes ist die Forderung, die man wirklich an die Gewichtsfunktion w stellt. Ist w stetig, so muss natürlich $w \geq 0$ sein und w darf nicht zu oft den Wert Null annehmen, denn dann könnte es ja $f \neq 0$ mit $I(f^2) = 0$ geben.

¹⁶⁵Der Einfachheit halber!

¹⁶⁶Der bei weitem wichtigste Fall ist dabei natürlich $w \equiv 1$.

¹⁶⁷Ein Funktional ist eine (lineare) Abbildung von einem Funktionenraum in einen Körper.

11.1 Quadraturformeln

Definition 11.1 Eine **Quadraturformel** Q_n mit den **Knoten**¹⁶⁸ x_0, \dots, x_n und Verwendung der **Gewichte** w_0, \dots, w_n ist ein lineares Funktional der Form

$$Q_n(f) = \sum_{j=0}^n w_j f(x_j), \quad f \in C[a, b]. \quad (11.3)$$

Eine Quadraturformel Q_n hat **Exaktheitsgrad** m für ein Integral I , falls

$$Q_n(p) = I(p), \quad p \in \Pi_m. \quad (11.4)$$

Die Forderung $f \in C[a, b]$ in (11.3) wird deswegen gemacht, daß die Punktauswertung überhaupt eine sinnvolle Aktion auf f ist (eine lediglich integrierbare Funktion kann man ja auf einer Nullmenge beliebig abändern).

Bemerkung 11.2 Oftmals fordert man, daß die Knoten der Quadraturformel alle im Intervall $[a, b]$ liegen und daß die Gewichte alle nichtnegativ sind. Im allgemeinen ist es aber nicht so einfach, diese beiden Forderungen zu erfüllen, ironischerweise passiert dies aber automatisch bei den "besten" Quadraturformeln, der sogenannten **Gauß-Quadratur**¹⁶⁹.

Definition 11.3 Die **Momente** μ_j , $j \in \mathbb{N}_0$, des Integrals I sind definiert als

$$\mu_j = I((\cdot)^j) = \int_a^b x^j w(x) dx.$$

Analog definiert man die Momente der Quadraturformel als $\gamma_{n,j} = Q_n((\cdot)^j)$.

Was bringt uns nun dieser Exaktheitsgrad? Auf alle Fälle einmal Abschätzungen der Approximationsgüte der Quadraturformel. Zu diesem Zweck definieren wir das **Fehlerfunktional**

$$E_n(f) = I(f) - Q_n(f), \quad f \in C[a, b],$$

und erhalten die folgende (einfache) Aussage.

Proposition 11.4 Ist Q_n eine Quadraturformel für I vom Exaktheitsgrad $m \geq 0$ mit nichtnegativen Gewichten und ist $f \in C^{m+1}[a, b]$, dann ist

$$|E_n(f)| \leq 2\mu_0 \frac{(b-a)^{m+1}}{(m+1)!} \left\| f^{(m+1)} \right\|_{\infty} \quad (11.5)$$

¹⁶⁸Schon wieder eine doppelte Benutzung desselben Begriffs, aber „Knoten“ ist in der Quadraturtheorie absolut Standard und alles andere wäre nur gekünstelt.

¹⁶⁹Dazu aber später noch mehr.

Beweis: Ist $p \in \Pi_m$ ein beliebiges Polynom, dann ist

$$\begin{aligned} |E_n(f)| &= |I(f) - Q_n(f)| = |I(f) - \underbrace{I(p) + Q_n(p)}_{=0} - Q_n(f)| = |I(f - p) - Q_n(f - p)| \\ &\leq |I(f - p)| + |Q_n(f - p)| \\ &\leq \int_a^b |f(x) - p(x)| w(x) dx + \sum_{j=0}^n w_j |f(x_j) - p(x_j)| \\ &\leq \underbrace{\left(\int_a^b w(x) dx + \sum_{j=0}^n w_j \right)}_{=2\mu_0} \|f - p\|_\infty, \end{aligned}$$

da

$$\mu_0 = I(1) = Q_n(1) = \sum_{j=0}^n w_j.$$

Wählt man p als das Interpolationspolynom an *irgendwelchen* Punkten¹⁷⁰ $\xi_0, \dots, \xi_m \in [a, b]$ und verwendet man die Fehlerformel

$$f(x) - L_m f(x) = \frac{f^{(m+1)}(\xi)}{(m+1)!} (x - \xi_0) \cdots (x - \xi_m), \quad x, \xi \in [a, b],$$

aus (8.24), so ergibt sich (11.5). □

Bemerkung 11.5 (Exaktheit)

1. Ist $[a, b] = [-1, 1]$, so kann man sogar die **Tschebyscheffknoten** zur Interpolation verwenden und erhält die wesentlich bessere Abschätzung

$$|E_n(f)| \leq 2\mu_0 \frac{1}{2^m (m+1)!} \|f^{(m+1)}\|_\infty.$$

Außerdem kann man ja immer das Intervall $[a, b]$ auf $[-1, 1]$ transformieren, indem man die Variable $t = 2\frac{x-a}{b-a} - 1$ einführt.

2. Der Beweis zeigt auch, daß

$$|E_n(f)| \leq 2\mu_0 \min_{p \in \Pi_m} \|f - p\|_\infty \rightarrow 0$$

für wachsendes m , nach dem Satz von Weierstraß, Satz 8.3.

¹⁷⁰Natürlich gilt: Je besser die Punkte, desto besser die Abschätzung.

3. Die einzige Konstante, die in (11.5) eingeht, ist μ_0 , ein Wert, der nur vom Integral selbst abhängt. Findet man also für ein vorgegebenes Integral I Quadraturformeln von hohem Exaktheitsgrad, dann kann man für "gute" Funktionen auch auf eine gute näherungsweise Berechnung des Integrals hoffen, vorausgesetzt, man erfüllt die Bedingung an die Nichtnegativität der Gewichte. Anderfalls erhielte man in (11.5) einen Term der Form

$$\sum_{j=0}^n |w_j|$$

und der hängt nun leider sehr wohl von der Quadraturformel ab und kann noch dazu sehr stark wachsen.

Gute Quadraturformeln wären nach Proposition 11.4 also solche, die nichtnegative Gewichte *und* hohen Exaktheitsgrad haben. Dabei gibt es natürlich Einschränkungen bezüglich der Anzahl der Knoten.

Proposition 11.6 (Exaktheitsgrad von Quadraturformeln) Für ein Integral I gilt:

1. Jede Quadraturformel Q_n für I mit $n + 1$ Knoten hat höchstens Exaktheitsgrad $2n + 1$.
2. Jede Quadraturformel Q_n für I mit $n + 1$ Knoten und Exaktheitsgrad $2n$ hat positive Gewichte.

Beweis: Sei $\omega(x) = (x - x_0) \cdots (x - x_n)$. Dann ist $\omega^2 \in \Pi_{2n+2}$ und es gilt¹⁷¹

$$I(\omega^2) = \int_a^b \omega^2(t) w(t) dt > 0 = \sum_{j=0}^n w_j \omega^2(x_j) = Q_n(\omega^2),$$

was 1. beweist. Für 2. verwenden wir eine Idee von Stieltjes¹⁷² und setzen

$$\ell_j(x) = \prod_{k \neq j} \frac{x - x_k}{x_j - x_k}, \quad j = 0, \dots, n, \quad x \in \mathbb{R},$$

das heißt, es ist $\ell_j \in \Pi_n$ und $\ell_j(x_k) = \delta_{jk}$, $j, k = 0, \dots, n$. Hat nun die Quadraturformel (mindestens) Exaktheitsgrad $2n$, dann ist

$$0 < \int_a^b \ell_j^2(t) w(t) dt = I(\ell_j^2) = Q_n(\ell_j^2) = \sum_{k=0}^n w_k \underbrace{\ell_j^2(x_k)}_{=\delta_{jk}} = w_j, \quad j = 0, \dots, n.$$

□

¹⁷¹Wäre $I(\omega^2) = 0$, dann wäre die "Gewichtsfunktion" w die Nullfunktion – diesen Trivialfall wollen wir ausschließen.

¹⁷²Siehe [10, S. 163]

Es gibt nun einen "einfachen" Weg, Quadraturformeln vom Exaktheitsgrad n zu bekommen, den wir auch im Beweis der vorhergehenden Proposition schon verwendet haben: Man verwendet **Interpolation** von f an den Punkten x_0, \dots, x_n durch ein Polynom vom Grad n , also

$$L_n f(x) = \sum_{j=0}^n f(x_j) \ell_j(x), \quad x \in \mathbb{R},$$

und setzt

$$\begin{aligned} Q_n(f) &= I(L_n f) = \int_a^b \left(\sum_{j=0}^n f(x_j) \ell_j(t) \right) w(t) dt = \sum_{j=0}^n \left(\int_a^b \ell_j(t) w(t) dt \right) f(x_j) \\ &= \sum_{j=0}^n \underbrace{I(\ell_j)}_{=: w_j} f(x_j). \end{aligned}$$

Die so erhaltenen Quadraturformeln bezeichnet man dann als **interpolatorisch**. Da $L_n : C[a, b] \rightarrow \Pi_n$ eine **Projektion** ist, ist $L_n p = p$ für alle $p \in \Pi_n$ und damit

$$Q_n(p) = I(L_n p) = I(p),$$

also hat Q_n (mindestens) **Exaktheitsgrad** n . Nun haben diese Quadraturformeln eben im allgemeinen keine nichtnegativen Gewichte mehr.

Definition 11.7 Sei I ein Integral und Q_n eine Quadraturformel dazu. Dann nennt man die Quadraturformel Q_n

1. **interpolatorisch**, wenn

$$w_j = I(\ell_j), \quad j = 0, \dots, n.$$

2. **Gauß-Formel**, wenn Q_n den maximalen Exaktheitsgrad $2n + 1$ hat.

Mit dieser Terminologie können wir wie in alle Quadraturformeln Q_n vom Exaktheitsgrad $\geq n$ charakterisieren, siehe [10, Theorem 3.2.1].

Satz 11.8 Seien I ein Integral und Q_n eine Quadraturformel dazu. Dann hat Q_n genau dann den Exaktheitsgrad $n + k$, $k = 0, \dots, n + 1$, wenn Q_n interpolatorisch ist und

$$0 = I(\omega \cdot p) = \int_a^b \omega(t) p(t) w(t) dt, \quad p \in \Pi_{k-1}, \quad \omega(x) = \prod_{j=0}^n (x - x_j). \quad (11.6)$$

Bemerkung 11.9 (Quadratur, Knoten und Exaktheit)

1. Gleichung (11.6) ist eine Bedingung an die **Knoten** x_j , $j = 0, \dots, n$, die passend gewählt werden müssen, um Exaktheit größer als n zu liefern – Exaktheitsgrad n ist ja bei allen interpolatorischen Formeln “eingebaut”. Mehr zu diesem Thema in Kapitel 11.4.
2. Wir können Satz 11.8 auch noch anders interpretieren: Es ist die Bestimmung der Knoten, was zählt, die Gewichte ergeben sich für jede “halbwegs gute” Quadraturformel dann einfach aus der Tatsache, daß diese interpolatorisch sein muss. Und genau das ist ja auch das Thema bei der Gauß–Quadratur: Wie bestimmt man die optimalen Knoten?

Beweis von Satz 11.8: Sei Q_n eine Quadraturformel vom Exaktheitsgrad $n + k \geq n$. Insbesondere ist also für $j = 0, \dots, n$,

$$I(\ell_j) = Q_n(\ell_j) = \sum_{k=0}^n w_k \underbrace{\ell_j(x_k)}_{=\delta_{jk}} = w_j,$$

also ist Q_n interpolatorisch. Außerdem ist¹⁷³ für jedes $p \in \Pi_{k-1}$ das Produkt¹⁷⁴ $p \cdot \omega$ in Π_{n+k} und verschwindet¹⁷⁵ an x_j , $j = 0, \dots, n$. Wegen des Exaktheitsgrads $k + n$ ist damit

$$I(\omega \cdot p) = Q_n(\omega \cdot p) = \sum_{j=0}^n w_j \underbrace{(\omega \cdot p)(x_j)}_{=0} = 0,$$

woraus auch (11.6) folgt.

Sei nun Q_n eine interpolatorische Quadraturformel für I , die (11.6) erfüllt und sei $p \in \Pi_{n+k}$. Wir schreiben p als¹⁷⁶ $p = q\omega + L_n p$, und erhalten, daß

$$I(p) = I(q \cdot \omega + L_n p) = \underbrace{I(q \cdot \omega)}_{=0} + \underbrace{I(L_n p)}_{=Q_n(L_n p)} = Q_n(p).$$

□

11.2 Klassische Formeln

Die “klassischen” Quadraturformeln für $w \equiv 1$ sind die sogenannten **Newton–Cotes–Formeln**, die man durch Polynominterpolation an *gleichverteilten*¹⁷⁷ Interpolationsknoten im Intervall $[a, b]$ erhält. Diese Konstruktionen gehen tatsächlich für den Fall $[a, b] = [-1, 1]$ und $w \equiv 1$ auf Newton (1687) bzw. Cotes¹⁷⁸ (1712) zurück. Setzt man

$$h = \frac{b-a}{n} \quad \text{und} \quad t = \frac{x-a}{h},$$

¹⁷³Mit der Konvention, daß $\Pi_{-1} = \{0\}$.

¹⁷⁴Zur Erinnerung: $\omega \in \Pi_{n+1}$

¹⁷⁵Ist $k = 0$, dann ist $p = p \cdot \omega = 0$.

¹⁷⁶Diese kleine Formel ist so einfach wie genial: Algebraisch gesehen ist Polynominterpolation also nichts anderes als der Rest bei Division durch das **Knotenpolynom** ω beziehungsweise, noch hochgestochener formuliert, die Normalform modulo des (Haupt-) Ideals $\langle \omega \rangle$.

¹⁷⁷Und diese Knotenwahl ist nicht gerade optimal!

¹⁷⁸Roger Cotes (1682–1716) bearbeitete die zweite Auflage von I. Newtons *Principia*.

dann wird $x \in [a, b]$ in $t \in [0, n]$ transformiert und es ist

$$\ell_j(x) = \prod_{k \neq j} \frac{x - \overbrace{(a + kh)}^{=x_k}}{a + jh - a - kh} = \prod_{k \neq j} \frac{a + th - a - kh}{h(j - k)} = \prod_{k \neq j} \frac{t - k}{j - k},$$

also, da $w \equiv 1$,

$$w_j = \int_a^b \ell_j(x) dx = h \int_0^n \prod_{k \neq j} \frac{t - k}{j - k} dt.$$

Man sieht sofort, daß diese Quadraturformeln *rationale* Gewichte haben also z.B. mit Maple exakt bestimmt und verwendet werden können. Die Newton–Cotes–Formeln kann man nun für kleine Werte von n explizit angeben – einige von ihnen haben sogar einen eigenen Namen; die folgende Tabelle stammt aus [41, S. 108], alle Gewichte sind immer noch mit der Intervalllänge $b - a$ zu multiplizieren.

n	Gewichte						Name	
1	$\frac{1}{2}$	$\frac{1}{2}$					Trapezregel	
2	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$				Simpson–Regel	
3	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$			“ Pulcherrima ”, 3/8–Regel	
4	$\frac{7}{90}$	$\frac{32}{90}$	$\frac{12}{90}$	$\frac{32}{90}$	$\frac{7}{90}$		Milne–Regel	
5	$\frac{19}{288}$	$\frac{75}{288}$	$\frac{50}{288}$	$\frac{50}{288}$	$\frac{75}{288}$	$\frac{19}{288}$		
6	$\frac{41}{840}$	$\frac{216}{840}$	$\frac{27}{840}$	$\frac{272}{840}$	$\frac{27}{840}$	$\frac{216}{840}$	$\frac{41}{840}$	Weddle–Regel

Die Liste bricht nicht ganz ohne tieferen Grund ab: Für $n > 6$ werden einige der Gewichte *negativ*.

Übung 11.1 Bestimmen Sie die Gewichte der Newton–Cotes–Formel für $n = 7$. \diamond

Die Simpson–Regel¹⁷⁹ ist auch als **Keplersche Faßregel** bekannt und besitzt, wenn man sich den Grad des zugehörigen Interpolationspolynome ansieht, zwar nur Exaktheitsgrad 2, aber eine Fehlerabschätzung der Form

$$|E_2 f| \leq \frac{h^5}{90} \left\| f^{(4)} \right\|_{\infty}, \quad f \in C^4[a, b], \quad (11.7)$$

die um eine Größenordnung besser ist als das, was man aus (11.5) erwarten könnte. Der Name “pulcherrima” bedeutet übrigens “die allerschönste”. Generell ist die Fehlerordnung der meisten Newton–Cotes–Formeln um einige Größenordnungen besser als man erwarten würde. Das hat einen relativ einfachen Grund ...

¹⁷⁹Nicht Bart oder Homer, sondern Thomas Simpson (1710–1761), Selfmademathematiker. Die Formel selbst war aber schon Cavalieri (1639) bekannt.

Beispiel 11.10 Die verbesserte Abschätzung (11.7) für die Simpson-Regel kann man beispielsweise durch Hermite-Interpolation beweisen. Dazu verwendet man für $x_0 = a$, $x_1 = a + h$ und $x_2 = b$ das Interpolationspolynom¹⁸⁰

$$H_3 f(x) = \underbrace{f(x_0) + (x - x_0) [x_0, x_2] f + (x - x_0)(x - x_2) [x_0, x_1, x_2] f}_{=L_2 f} + (x - x_0)(x - x_1)(x - x_2) [x_0, x_1, x_1, x_2] f,$$

das die Bedingungen

$$H_3 f(x_j) = f(x_j), \quad j = 0, 1, 2, \quad (H_3 f)'(x_1) = f'(x_1)$$

erfüllt. Da

$$\int_a^b (x - x_0)(x - x_1)(x - x_2) dx = 0,$$

ist

$$\int_a^b H_3 f(t) dt = \int_a^b L_2 f(t) dt = Q_2(f),$$

aber für $f \in C^4[a, b]$ liefert $H_3 f$ den Fehlerterm

$$\begin{aligned} I - Q_2(f) &= \int_a^b (f - H_3(f))(t) dt = \int_a^b (t - x_0)(t - x_1)^2(t - x_2) \frac{f^{(4)}(\xi(t))}{4!} dt \\ &= \frac{f^{(4)}(\xi)}{4!} \int_0^{2h} t(t-h)^2(t-2h) dt = -\frac{h^5}{90} f^{(4)}(\xi), \end{aligned}$$

also die Größenordnung h^5 statt h^3 .

Warum man sich für den Exponenten von h interessiert sollte klar sein: Wenn das Intervall klein wird, dann fällt der Fehler viel schneller und die Formel ist genauer. Allerdings muss f auch „glatter“ sein und eine stetige vierte Ableitung besitzen.

11.3 Zusammengesetzte Quadraturformeln

Bereits in der (sehr groben) Fehlerabschätzung (11.5) zeigt sich, daß eine Quadraturformel auf einem *kleinen* Intervall normalerweise genauer sein wird, als auf einem großen Intervall. Die Idee ist also einfach, nur die Notation ein wenig aufwendiger. Sei also Q_n eine Quadraturformel, dann definieren wir zu $N \in \mathbb{N}$ die aufsteigende Punktfolge $x_{j,k}$, $j = 1, \dots, N$, $k = 0, \dots, n$, wobei $x_{j,k} < x_{j',k'}$ falls $j < j'$ oder $j = j'$ und $k < k'$, sowie

$$x_{1,0} = a, \quad x_{j,n} = x_{j+1,0}, \quad j = 1, \dots, N-1, \quad x_{N,n} = b,$$

so daß wir eine *überlappende* Zerlegung von $[a, b]$ erhalten. Das Teilintegral auf $[x_{j,0}, x_{j,n}]$

¹⁸⁰Die etwas seltsame Reihenfolge x_0, x_2, x_1, x_1 , in der die Interpolationspunkte in die Newton-Darstellung eingegeben werden, ist kein Tipfehler, sondern durchaus Absicht.

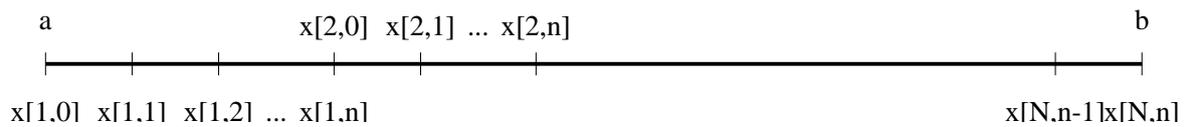


Abbildung 11.1: Zerlegung des Intervalls $[a, b]$ für zusammengesetzte Quadraturformel.

bestimmt man nun jeweils mit der transformierten Quadraturformel Q_n – das setzt natürlich voraus, daß die Knoten $x_{j,k}$ ebenfalls *Transformationen* der Knoten der Quadraturformel Q_n sind. Genauer: Bezeichnet man mit

$$0 = x_0 < x_1 < \dots < x_{n-1} < x_n = 1$$

die Knoten von Q_n , dann ist

$$x_k = \frac{x_{j,k} - x_{j,0}}{x_{j,n} - x_{j,0}}, \quad j = 1, \dots, N. \quad (11.8)$$

Ausgehend von einer Quadraturformel Q_n für $I = \int_0^1 dt$ ergibt sich dann die zusammengesetzte Formel als

$$Q_{n,N}(f) = \sum_{j=1}^N \underbrace{(x_{j,n} - x_{j,0})^{-1}}_{=: h_j^{-1}} Q_n \left(f \left(\frac{\cdot - x_{j,0}}{x_{j,n} - x_{j,0}} \right) \right) = \sum_{j=1}^N h_j^{-1} Q_n \left(f \left(h_j^{-1}(\cdot - x_{j,0}) \right) \right). \quad (11.9)$$

Beispiel 11.11 *Im Newton–Cotes–Fall wäre*

$$x_{j,k} = a + \frac{(j-1)(n+1) + k}{Nn} (b-a) = a + h((j-1)(n+1) + k), \quad h = \frac{b-a}{Nn}.$$

Damit entspricht die zusammengesetzte Trapezregel gerade der Integration des stückweise linearen Interpolanten.

Das Ziel ist es also, das Integral

$$I(f) = \int_a^b f(t) dt = \sum_{j=1}^N \int_{x_{j,0}}^{x_{j,n}} f(t) dt =: \sum_{j=1}^N I_j(f), \quad f \in C[a, b],$$

anzunähern und eine Abschätzung für den Fehler anzugeben.

Proposition 11.12 Sei Q_n die Newton–Cotes–Formel für das Integral $\int_0^1 dt$ und $Q_{n,N}$ die daraus gebildete zusammengesetzte Quadraturformel. Dann ist, für $f \in C^{n+1}[a, b]$,

$$|I(f) - Q_{n,N}(f)| \leq (b - a) \max_{j=1, \dots, n} \left(h_j^n \frac{\|f^{(n+1)}\|_{[x_{j,0}, x_{j,n}]}}{n!} \right). \quad (11.10)$$

Bemerkung 11.13 Aus (11.10) folgt sofort die globale Fehlerabschätzung

$$|I(f) - Q_{n,N}(f)| \leq (b - a) \left(\max_{j=1, \dots, n} h_j \right)^n \frac{\|f^{(n+1)}\|_{[a,b]}}{n!}, \quad (11.11)$$

aber dennoch sagt uns (11.10) einiges mehr: Man könnte versuchen, den Gesamtfehler dadurch zu drücken, daß man h_j klein macht, wo der Betrag von $f^{(n+1)}$ sehr groß wird und dafür die Schrittweite in Bereichen groß macht, wo $f^{(n+1)}$ betragsmäßig sehr klein wird. Auf diese Art und Weise erhält man, unter Ausnutzung von zusätzlicher¹⁸¹ Information über f möglicherweise bessere Quadraturformeln, ohne die Zahl der Knoten und damit den Rechenaufwand zu vergrößern.

Derartige Verfahren, die versuchen, ihre Parameter (in diesem Fall die Knoten) aus Informationen über die zu verarbeitenden Daten (in diesem Fall die Funktion, bzw. deren Werte an bestimmten Stellen) zu bestimmen, bezeichnet man als **adaptiv**. Dies steht im Gegensatz zu (nichtadaptiven) Verfahren, bei denen die Parameter von Anfang an feststehen – solche Verfahren sind zwar meist etwas schneller, dafür aber weniger flexibel.

Beweis von Proposition 11.12: Mit der Bezeichnung

$$t_{j,k} := \frac{x_{j,k} - x_{j,0}}{x_{j,n} - x_{j,0}}, \quad j = 1, \dots, N,$$

ist, für $j = 1, \dots, N$,

$$\begin{aligned} I_j(f) &= \int_{x_{j,0}}^{x_{j,n}} f(t) dt = \int_{x_{j,0}}^{x_{j,n}} (L_n f(t) + [t, x_{j,0}, \dots, x_{j,n}] f) dt \\ &= \sum_{k=0}^n h_j^{-1} \underbrace{\int_0^1 \ell_k(h^{-1}(t + x_{j,k})) dt}_{=w_k} f(x_{j,k}) + \int_{x_{j,0}}^{x_{j,n}} [t, x_{j,0}, \dots, x_{j,n}] f dt \\ &= h_j^{-1} Q_{n,N} \left(f \left(h_j^{-1}(\cdot - x_{j,0}) \right) \right) + \underbrace{\int_{x_{j,0}}^{x_{j,n}} (t - x_{j,0}) \cdots (t - x_{j,n}) \frac{f^{(n+1)}}{(n+1)!} (\xi(t)) dt}_{=:R_j(f)}. \end{aligned}$$

¹⁸¹Stellt sich natürlich die Frage, wo man diese herbekommt.

Damit ist

$$I(f) = \sum_{j=1}^N I_j(f) = \sum_{j=1}^N h_j^{-1} Q_n \left(f \left(h_j^{-1} (\cdot - x_{j,0}) \right) \right) + \sum_{j=1}^N R_j(f) = Q_{n,N}(f) + \sum_{j=1}^N R_j(f).$$

Den Fehlerterm $R_j(f)$ spalten wir wie folgt auf und verwenden den Mittelwertsatz der Integration¹⁸²

$$\begin{aligned} |R_j(f)| &= \left| \sum_{k=0}^{n-1} \int_{x_{j,k}}^{x_{j,k+1}} (x - x_{j,0}) \cdots (x - x_{j,n}) \frac{f^{(n+1)}(\xi(x))}{(n+1)!} dx \right| \\ &= \left| \sum_{k=0}^{n-1} \frac{f^{(n+1)}(\xi_k)}{(n+1)!} \int_{x_{j,k}}^{x_{j,k+1}} (x - x_{j,0}) \cdots (x - x_{j,n}) dx \right| \leq n h_j^{n+1} \frac{\|f^{(n+1)}\|_{[x_{j,0}, x_{j,n}]}}{(n+1)!} \\ &\leq h_j^{n+1} \frac{\|f^{(n+1)}\|_{[x_{j,0}, x_{j,n}]}}{n!}. \end{aligned}$$

Damit ist

$$|I - Q_{n,N}(f)| \leq \sum_{j=1}^n |R_j(f)| \leq \left(\max_{j=1, \dots, n} h_j^n \frac{\|f^{(n+1)}\|_{[x_{j,0}, x_{j,n}]}}{n!} \right) \underbrace{\sum_{j=1}^n h_j}_{=b-a}.$$

□

Beispiel 11.14 Mit der verbesserten Fehlerformel aus Beispiel 11.10 und den Punkten $x_j = a + jh$, $j = 0, \dots, 2N$, können wir die den Fehler für die **zusammengesetzten Simpson-Regel** bestimmen:

$$\begin{aligned} E_2(f) &= - \sum_{j=0}^{2N-2} \frac{1}{90} h^5 f^{(4)}(\xi_j) = - \frac{1}{90} h^4 \sum_{j=0}^{2N-2} \underbrace{\frac{b-a}{2N-1}}_{=h} f^{(4)}(\xi_j) = - \frac{b-a}{90} h^4 \sum_{j=0}^{2N-2} \frac{f^{(4)}(\xi_j)}{2N-1} \\ &= - \frac{b-a}{90} h^4 f^{(4)}(\xi), \quad \xi \in \left[\min_{j=0, \dots, 2N-2} \xi_j, \max_{j=0, \dots, 2N-2} \xi_j \right]. \end{aligned} \quad (11.12)$$

Die Existenz der "magischen" Stelle ξ ergibt sich dabei aus der Ungleichung

$$\min_{j=0, \dots, 2N-2} f^{(4)}(\xi_j) \leq \sum_{j=0}^{2N-2} \frac{f^{(4)}(\xi_j)}{2N-1} \leq \max_{j=0, \dots, 2N-2} f^{(4)}(\xi_j)$$

¹⁸²Dafür darf das Polynom keinen Vorzeichenwechsel haben – deswegen die Aufspaltung.

für die in der Summe gebildete **Konvexkombination** und nach dem Zwischenwertsatz wird deren Wert eben an einer Stelle ξ , angenommen, die im kleinsten Intervall liegt, das alle ξ_j , $j = 0, \dots, 2N - 2$, enthält.

Diese Fehlerabschätzung (11.12) ist wesentlich besser als die aus Proposition 11.12, vorausgesetzt natürlich, daß f hinreichend oft differenzierbar ist und daß sich die vierte Ableitung von f im Vergleich zur zweiten Ableitung "brav verhält"¹⁸³, sonst muß man h schon sehr klein wählen.

Geometrisch ist die zusammengesetzte Simpsonformel

$$Q_{2,N}(f) = \frac{b-a}{6N} (f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \dots + 4f_{2N-1} + f_{2N}),$$

$$f_j = f(a + jh), \quad j = 0, \dots, 2N, \quad h = \frac{b-a}{2N},$$

die Integration der Interpolationsparabeln an den Punkten

$$x_{2j}, x_{2j+1}, x_{2j+2}, \quad j = 0, \dots, 2N - 2,$$

siehe Abb. 11.2.

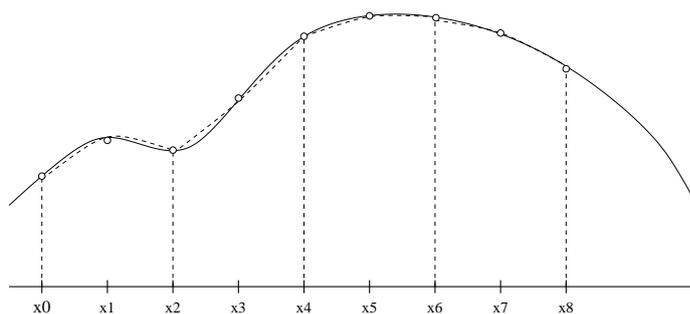


Abbildung 11.2: Geometrische Interpretation der zusammengesetzten Simpson-Formel.

11.4 Gauß-Quadratur

Eine **Gauß-Formel** ist eine Quadraturformel Q_n , die den *maximalen* Exaktheitsgrad $2n + 1$ hat. Die Existenz solcher Formeln und ihre Konstruktion wurden bereits 1816 von Gauß [9] angegeben. Das Originalargument von Gauß verwendet überraschenderweise ganz andere Techniken und findet sich ebenfalls im Anhang

¹⁸³Daß das nicht der Fall sein muß zeigen schon die Monome x^m , m groß, bei denen die vierte Ableitung um einen Faktor $(m-2)(m-3) = O(m^2)$ größer ist – soviel zum Wert "asymptotischer" Blankoaussagen.

Wir können uns aber, dank Satz 11.8, das Leben wesentlich leichter machen, denn wir wissen, daß eine Quadraturformel mit Knoten x_j , $j = 0, \dots, n$, genau dann eine Gaußsche Quadraturformel ist, wenn

1. sie interpolatorisch ist.
2. das Polynom $\omega = (\cdot - x_0) \cdots (\cdot - x_n)$ zu allen Polynomen vom Grad $\leq n$ **orthogonal** ist, das heißt, wenn

$$\int_a^b p(x) \omega(x) dx = 0, \quad p \in \Pi_n.$$

Damit gehört ω aber zu einer ganz bestimmten und besonders interessanten Klasse von Polynomen.

Definition 11.15 Ein Polynom $p \in \Pi_n$ heißt **orthogonales Polynom** der Ordnung n bezüglich der Gewichtsfunktion w , wenn $p \neq 0$ und

$$\int_a^b p(x) q(x) w(x) dx = 0, \quad q \in \Pi_{n-1}.$$

Das heißt, daß p orthogonal zu Π_{n-1} ist, und zwar in Bezug auf das **Skalarprodukt**

$$\langle f, g \rangle := \langle f, g \rangle_w := \int_a^b f(x) g(x) w(x) dx. \quad (11.13)$$

Übung 11.2 Zeigen Sie: Ist w eine stetige, nichtnegative Funktion und ist $w(x) > 0$ für ein $x \in [a, b]$, dann ist $\langle f, g \rangle_w$ ein Skalarprodukt auf $\Pi \times \Pi$. Kann man eine der obigen Bedingungen an w weglassen? \diamond

Nun kann man leicht sehen, daß es für "vernünftige" Gewichtsfunktionen immer orthogonale Polynome *aller* Ordnungen $n \in \mathbb{N}_0$ gibt. Man erhält diese Polynome zum Beispiel durch das **Gram-Schmidt Orthogonalisierungsverfahren**: Sind p_0, \dots, p_n orthogonale Polynome der Ordnung $0, \dots, n$ und ist $w \geq 0$ und *stetig*¹⁸⁴, dann ist

$$p_{n+1}(x) = x^{n+1} - \sum_{j=0}^n \frac{\langle (\cdot)^{n+1}, p_j \rangle}{\langle p_j, p_j \rangle} p_j(x), \quad x \in \mathbb{R}, \quad (11.14)$$

ein orthogonales Polynom der Ordnung $n + 1$, denn es ist, für $k = 0, \dots, n$,

$$\langle p_{n+1}, p_k \rangle = \left\langle (\cdot)^{n+1}, p_k \right\rangle - \sum_{j=0}^n \frac{\langle (\cdot)^{n+1}, p_j \rangle}{\langle p_j, p_j \rangle} \underbrace{\langle p_j, p_k \rangle}_{=\delta_{jk}} = \left\langle (\cdot)^{n+1}, p_k \right\rangle - \left\langle (\cdot)^{n+1}, p_k \right\rangle = 0.$$

¹⁸⁴Das ist sogar mehr als wirklich nötig wäre, es genügt, wenn das Integral $I = \int_a^b w(x) dx$ *strikt* *quadratpositiv* ist, d.h. $I(f^2) > 0$ für alle $f \in \Pi$.

Zusammen mit Übung 11.3 zeigt dies, daß p_{n+1} ein orthogonales Polynom der Ordnung $n+1$ ist¹⁸⁵. Da der **Leitterm** von p_{n+1} bei dieser Konstruktion das **Monom** x^{n+1} ist, bezeichnet man die durch (11.14) induktiv konstruierten orthogonalen Polynome als **monische orthogonale Polynome**.

Übung 11.3 Zeigen Sie, daß die orthogonalen Polynome p_0, \dots, p_n eine Basis von Π_n bilden. \diamond

Damit könnte man also Gaußsche Quadraturformeln dadurch konstruieren, daß man mittels (11.14) eine Folge p_0, \dots, p_n von monischen orthogonalen Polynomen konstruiert¹⁸⁶ und dann mit einem geeigneten Verfahren die Nullstellen von p_n bestimmt – daß die alle einfach und reell sind, werden wir noch sehen, und zwar in Proposition 11.17.

Ein etwas netterer Ansatz verwendet lineare Algebra und die **Momentenmatrix**¹⁸⁷

$$M = \left[\left\langle (\cdot)^j, (\cdot)^k \right\rangle : j, k = 0, \dots, n \right] = \begin{bmatrix} \mu_0 & \mu_1 & \dots & \mu_n \\ \mu_1 & \mu_2 & \dots & \mu_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_n & \mu_{n+1} & \dots & \mu_{2n} \end{bmatrix}.$$

Diese $(n+1) \times (n+1)$ -Matrix ist positiv definit,

$$x^T M x = \left\langle \sum_{j=0}^n x_j (\cdot)^j, \sum_{k=0}^n x_k (\cdot)^k \right\rangle =: \langle p, p \rangle = I(p^2) > 0, \quad x \in \mathbb{R}^{n+1} \setminus \{0\},$$

und besitzt eine QR-Zerlegung $M = QR$, bei der alle Diagonalelemente von R ungleich Null sind, denn R ist ja invertierbar. Daher ist, für $j = 0, \dots, n$,

$$\begin{aligned} r_{nn} \delta_{nj} &= r_{nj} = (Q^T M)_{nj} = \sum_{k=0}^n \underbrace{q_{kn}}_{=(Q^T)_{nk}} m_{kj} = \sum_{k=0}^n q_{kn} \left\langle (\cdot)^k, (\cdot)^j \right\rangle \\ &= \left\langle \sum_{k=0}^n q_{kn} (\cdot)^k, (\cdot)^j \right\rangle, \end{aligned}$$

das heißt, das Polynom

$$q_n(x) = \sum_{k=0}^n q_{kn} x^k, \quad x \in \mathbb{R}, \quad q_n \in \Pi_n,$$

ist orthogonal zu Π_{n-1} .

¹⁸⁵... das bis auf Normierung eindeutig bestimmt ist.

¹⁸⁶Nur zur Erinnerung: für die Berechnung von p_k braucht man p_0, \dots, p_{k-1} .

¹⁸⁷Eine Matrix, deren Einträge auf den "Antidiagonalen" konstant sind, bezeichnet man als **Hankel-matrix** zu dem Vektor (μ_0, \dots, μ_{2n}) .

Übung 11.4 Untersuchen Sie die Beziehung zwischen der klassischen Gram–Schmidt Orthogonalisierung und der QR–Zerlegung. \diamond

Wenn wir jetzt mal für einen Moment¹⁸⁸ annehmen, daß Gaußsche Quadraturformeln für alle “vernünftigen” Gewichtsfunktionen existieren, dann können wir mit den inzwischen vertrauten Methoden auch den Fehler angeben.

Satz 11.16 Sei $w \in C[a, b]$ eine stetige, nichtnegative Gewichtsfunktion und Q_n^* die Gauß–Quadraturformel für das Integral $I = \int_a^b w(x) dx$. Dann gibt es für jedes $f \in C^{2n+2}[a, b]$ ein $\xi \in [a, b]$, so daß

$$I(f) - Q_n^*(f) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \langle \omega_{n+1}, \omega_{n+1} \rangle, \quad (11.15)$$

wobei $\omega_{n+1} = (\cdot - x_0) \cdots (\cdot - x_n)$ das monische orthogonale Polynom der Ordnung $n+1$ ist.

Beweis: Aus den vorherigen Bemerkungen wissen wir, daß die Knoten der Gaußschen Quadraturformel¹⁸⁹ gerade die Nullstellen von ω_{n+1} sind. Nun sei $H_n \in \Pi_{2n+1}$ wieder der **Hermite–Interpolant**, der die Bedingungen

$$H_n f(x_j) = f(x_j), \quad (H_n f)'(x_j) = f'(x_j), \quad j = 0, \dots, n,$$

erfüllt. Wegen der Exaktheit der Gauß–Formel ist dann

$$I(H_n f) = Q_n^*(H_n f) = \sum_{j=0}^n w_j \underbrace{H_n f(x_j)}_{=f(x_j)} = \sum_{j=0}^n w_j f(x_j) = Q_n^*(f),$$

also

$$I(f) - Q_n^*(f) = I(f) - I(H_n f) = I(f - H_n f).$$

Für $f - H_n f$ verwenden wir wieder die Fehlerformel¹⁹⁰

$$(f - H_n f)(x) = \frac{f^{(2n+2)}(\xi(x))}{(2n+2)!} (x - x_0)^2 \cdots (x - x_n)^2 = \frac{f^{(2n+2)}(\xi(x))}{(2n+2)!} \omega_{n+1}^2(x),$$

und mit dem Mittelwertsatz der Integralrechnung erhalten wir, daß

$$\begin{aligned} I(f - H_n f) &= \int_a^b \frac{f^{(2n+2)}(\xi(x))}{(2n+2)!} \omega_{n+1}^2(x) w(x) dx = \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \int_a^b \omega_{n+1}^2(x) w(x) dx \\ &= \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \langle \omega_{n+1}, \omega_{n+1} \rangle. \end{aligned}$$

□

¹⁸⁸Genauer: bis zum Beweis von Proposition 11.17.

¹⁸⁹Immer vorausgesetzt, diese existiert, aber ...

¹⁹⁰Siehe auch Beispiel 11.10.

11.5 Orthogonale Polynome

Wir haben gesehen, daß die Existenz und Bestimmung Gaußscher Quadraturformeln wesentlich mit orthogonalen Polynomen verknüpft ist. Das ist sicherlich eine, aber nicht die einzige, Begründung, sich orthogonale Polynome ein bißchen näher anzusehen. Zuerst rechtfertigen wir einmal die Existenz (und Eindeutigkeit) Gaußscher Quadraturformeln, indem wir zeigen, daß ein orthogonales Polynom immer einfache, reelle Nullstellen haben muß.

Proposition 11.17 *Ein orthogonales Polynom $p_n \in \Pi_n$ vom Grad n zum Integral $\int_a^b w(x) dx$ bezüglich einer nichtnegativen, stetigen Gewichtsfunktion $w \in C[a, b]$ hat n einfache, reelle Nullstellen in $[a, b]$.*

Beweis: Nehmen wir zuerst an, irgendein p_n , $n \geq 1$, hätte *keine* Nullstelle in $[a, b]$, sei also beispielsweise $p_n > 0$. Dann ist

$$0 < \int_a^b p_n(x) w(x) dx = \langle 1, p_n \rangle,$$

im Widerspruch zur Orthogonalität. Hat p_n eine Nullstelle der Ordnung ≥ 2 in $[a, b]$, sagen wir ξ , dann ist $p_n / (x - \xi)^2 \in \Pi_{n-2}$ ein Polynom, das überall genau dasselbe Vorzeichen hat wie p_n . Wegen der Orthogonalität ist dann

$$0 = \left\langle p_n, \frac{p_n}{(\cdot - \xi)^2} \right\rangle = \left\langle 1, \frac{p_n^2}{(\cdot - \xi)^2} \right\rangle,$$

woraus $p_n = 0$ folgen würde – auch wieder nicht das, was man sich unter einem “anständigen” orthogonalen Polynom vorstellt.

Also hat jedes p_n in $[a, b]$ nur *einfache* Nullstellen, sagen wir ξ_1, \dots, ξ_m , $m \leq n$. Nun ist aber das Polynom

$$p_n(x) \underbrace{(x - \xi_1) \cdots (x - \xi_m)}_{=: q \in \Pi_m}$$

auf $[a, b]$ entweder ≥ 0 oder ≤ 0 und damit

$$0 \neq \int_a^b p_n(x) q(x) w(x) dx = \langle p_n, q \rangle,$$

weswegen $m = n$ sein muß, denn ansonsten wäre das Integral wegen der Orthogonalität von p_n ja $= 0$. \square

Aus Satz 11.8 und den Betrachtungen über die Gaußschen Quadraturformeln erhalten wir nun die Existenz und Eindeutigkeit dieser speziellen Formeln.

Korollar 11.18 Sei $w \in C[a, b]$ eine positive¹⁹¹ Gewichtsfunktion. Dann gibt es genau eine Quadraturformel Q_n vom Exaktheitsgrad $2n + 1$, deren Knoten die Nullstellen des¹⁹² orthogonalen Polynoms sind.

Die nächste Eigenschaft ist eine **Rekursionsformel** für orthonormale Polynome, der Beweis ist aus [8, S. 234].

Satz 11.19 Die orthonormalen¹⁹³ Polynome p_n bezüglich einer positiven Gewichtsfunktion $w \in C[a, b]$ erfüllen eine **Drei-Term-Rekursionsformel**

$$p_n(x) = (\alpha_n x + \beta_n) p_{n-1}(x) - \gamma_n p_{n-2}(x), \quad x \in \mathbb{R}, \quad n \in \mathbb{N}, \quad (11.16)$$

mit Startwerten

$$p_0 \equiv \mu_0^{-1/2}, \quad p_{-1} \equiv 0. \quad (11.17)$$

Beweis: Wir zeigen, daß die Polynome q_n , definiert durch

$$q_{n+1}(x) = x p_n(x) - \underbrace{\langle (\cdot) p_n, p_n \rangle}_{=: a_n} p_n(x) - \underbrace{\sqrt{\langle q_n, q_n \rangle}}_{=: b_n > 0} p_{n-1}(x), \quad x \in \mathbb{R}, \quad n \in \mathbb{N}_0,$$

mit den Startwerten wir in (11.17) orthogonale Polynome sind. Die orthonormalen Polynome sind dann

$$\begin{aligned} p_{n+1}(x) &= \frac{q_{n+1}(x)}{\sqrt{\langle q_{n+1}, q_{n+1} \rangle}} \\ &= \left(\frac{1}{\sqrt{\langle q_{n+1}, q_{n+1} \rangle}} x - \frac{\langle (\cdot) p_n, p_n \rangle}{\sqrt{\langle q_{n+1}, q_{n+1} \rangle}} \right) p_n(x) - \sqrt{\frac{\langle q_n, q_n \rangle}{\langle q_{n+1}, q_{n+1} \rangle}} p_{n-1}(x), \end{aligned}$$

woraus (11.16) sofort folgt.

Die Orthogonalität selbst beweisen wir durch Induktion über n ; für $n = 0$, das heißt $q_0 \equiv 1$, ist nichts zu zeigen und $n = 1$ ist auch offensichtlich. Sei also die Orthogonalität für ein $n \geq 1$ bewiesen, dann betrachten wir, für $j = 0, \dots, n$, die Werte

$$\langle q_{n+1}, p_j \rangle = \langle (\cdot) p_n, p_j \rangle - a_n \langle p_n, p_j \rangle - b_n \langle p_{n-1}, p_j \rangle;$$

für $j < n - 1$ sind diese Werte natürlich Null. Unter Verwendung der Induktionshypothese ist außerdem

$$\begin{aligned} \langle q_{n+1}, p_{n-1} \rangle &= \langle (\cdot) p_n, p_{n-1} \rangle - b_n \langle p_{n-1}, p_{n-1} \rangle = \langle p_n, (\cdot) p_{n-1} \rangle - b_n \langle p_{n-1}, p_{n-1} \rangle \\ &= \langle p_n, q_n \rangle + \underbrace{a_{n-1} \langle p_n, p_{n-1} \rangle}_{=0} + \underbrace{b_{n-1} \langle p_n, p_{n-2} \rangle}_{=0} - \underbrace{b_n \langle p_{n-1}, p_{n-1} \rangle}_{=1} \\ &= \left\langle p_n, \sqrt{\langle q_n, q_n \rangle} p_n \right\rangle - b_n = \underbrace{\sqrt{\langle q_n, q_n \rangle}}_{=b_n} \underbrace{\langle p_n, p_n \rangle}_{=1} - b_n = 0, \end{aligned}$$

¹⁹¹Das heißt: $w \geq 0$ und es gibt mindestens ein $x \in [a, b]$, so daß $w(x) > 0$.

¹⁹²Normierung spielt ja für die Lage der Nullstellen keine Rolle.

¹⁹³Dadurch sind die Polynome bis auf Vorzeichen *eindeutig* definiert, in mehreren Variablen stimmt das übrigens nicht mehr – nur ein Beispiel, warum es da noch einiges zu tun gibt, siehe [48].

sowie

$$\langle q_{n+1}, p_n \rangle = \underbrace{\langle x p_n, p_n \rangle}_{=a_n} - a_n \underbrace{\langle p_n, p_n \rangle}_{=1} = 0,$$

womit also $q_{n+1} \perp \Pi_n$. Daß $q_{n+1} \neq 0$ ist, folgt induktiv aus der Tatsache, daß $p_n \neq 0$.
□

Über die Rekursionsformel könnte man nun sehr elegant die Knoten und Gewichte der Quadraturformel bestimmen, leider braucht man dazu aber Eigenwertverfahren und für die war nun leider absolut keine Zeit mehr. Insofern:

*Manches sagt ich,
mehr noch wollt ich,
ließe zur Rede
Raum das Geschick.
Die Stimme weicht,
Wunden schwellen:
Wahres sprach ich;
will nun enden.*

(Die Edda, Das jüngere Sigurdlied)

*And it didn't stop being magic just
because you found out how it was done*

T. Pratchett, „Wee Free Men“

Anhang | A

In diesem Abschnitt sind Beweise und anderes vertiefendes Material zusammengefasst, die zum Verständnis des Stoffes beitragen, aber der Endlichkeit der Zeit zum Opfer fielen, die in einem Semester zur Verfügung steht. Es ist also nicht verboten, sich die Sachen auch mal anzusehen.

A.1 Rückwärtsfehler der Gauß–Elimination

Für den Beweis der Rückwärtsfehlerabschätzung in Satz 4.21 sehen wir uns zuerst einmal den Fehler bei der Vorwärtselimination und Rücksubstitution an. Dazu wird es opportun sein, auch das *alternative* Modell der Gleitkommaarithmetik zu verwenden, das annimmt, daß

$$x \odot y = \frac{x \cdot y}{1 + \delta}, \quad |\delta| \leq \hat{u}, \quad \cdot = +, -, \times, /. \quad (\text{A.1})$$

Diese Annahme ist (ohne Beweis!) durchaus sinnvoll, denn mit $\hat{u} = 4u$ kann man auf alle Fälle garantieren, daß (2.4) und (A.1) gleichzeitig gelten.

Übung A.1 Zeigen Sie:

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

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

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 A.1 Die durch Vorwärtselimination (Rücksstitution) 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) Dreiecksmatrix ist, die

$$|\hat{L} - L| \leq n\hat{u} |L| + O(\hat{u}^2), \quad (\text{A.2})$$

beziehungsweise

$$|\hat{U} - U| \leq n\hat{u} |U| + O(\hat{u}^2), \quad (\text{A.3})$$

erfüllt.

Beweis: Wir bezeichnen mit $\ell^j = [\ell_{jk} : k = 1, \dots, j-1] \in \mathbb{R}^{j-1}$ den Anfang des j -ten Zeilenvektors von L ¹⁹⁴. Mit der **Vorwärtselimination** bestimmen wir nun Vektoren $\hat{x}^j \in \mathbb{R}^j$, $j = 1, \dots, n$, durch

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

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

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

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

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

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

$$|\hat{L} - L| \leq \max\{2, n-1\} \hat{u} |L| + O(\hat{u}^2) \quad (\text{A.4})$$

gilt, also insbesondere (A.2).

Der Beweis für die **Rücksstitution** funktioniert natürlich ganz analog. \square

Übung A.2 Geben Sie ein Verfahren an, das die Vorwärtselimination bzw. Rücksstitution mit einem von n unabhängigen Rückwärtsfehler ausführt. \diamond

Bemerkung A.2 Gehen wir nur vom Standardmodell (2.4) aus, dann ist (A.4) mit der Konstante $\max\{4, n-1\}$ erfüllt, was für $n \geq 5$ auch kein Beinbruch ist.

¹⁹⁴Also das, was unterhalb der Diagonalen steht

Proposition A.3 Sei $A \in \mathbb{F}^{n \times n}$. Wenn alle Hauptminoren von A invertierbar sind, dann erfüllt die berechnete LU-Zerlegung \hat{L}, \hat{U} die Bedingung

$$|A - \hat{L}\hat{U}| \leq 3n\hat{u} (|A| + |\hat{L}| |\hat{U}|) + O(\hat{u}^2). \quad (\text{A.5})$$

Beweis: Induktion über n . Im Fall $n = 1$ ist $\hat{U} = A, \hat{L} = 1$, und (A.5) ist trivialerweise richtig: Die linke Seite hat Wert 0.

Sei also (A.5) für ein $n \geq 1$ bewiesen und sei $A \in \mathbb{R}^{n+1 \times n+1}$, das wir schreiben als

$$A = \left[\begin{array}{c|c} \mathbf{a}_{11} & \mathbf{w}^T \\ \mathbf{v} & B \end{array} \right], \quad \mathbf{v}, \mathbf{w} \in \mathbb{R}^n, B \in \mathbb{R}^{n \times n}. \quad (\text{A.6})$$

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

$$|\hat{\mathbf{y}}^{(1)} - \mathbf{y}^{(1)}| \leq \hat{u} |\mathbf{y}^{(1)}| = \hat{u} \frac{|\mathbf{v}|}{|\mathbf{a}_{11}|}$$

sowie

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

also

$$|\hat{A}_1 - A_1| \leq 3\hat{u} |A_1| + O(\hat{u}^2). \quad (\text{A.7})$$

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

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

Wir erhalten dann, daß

$$\hat{L} = \left[\begin{array}{c|c} 1 & 0 \\ \hat{\mathbf{y}}^{(1)} & \hat{L}_1 \end{array} \right], \quad \hat{U} = \left[\begin{array}{c|c} \mathbf{a}_{11} & \mathbf{w}^T \\ 0 & \hat{U}_1 \end{array} \right],$$

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

$$\begin{aligned} |A - \hat{L}\hat{U}| &= \left| \left[\begin{array}{c|c} \mathbf{a}_{11} & \mathbf{w}^T \\ \mathbf{v} & B \end{array} \right] - \left[\begin{array}{c|c} \mathbf{a}_{11} & \mathbf{w}^T \\ \mathbf{a}_{11} \hat{\mathbf{y}}^{(1)} & \hat{\mathbf{y}}^{(1)} \mathbf{w}^T + \hat{L}_1 \hat{U}_1 \end{array} \right] \right| \\ &= \left| \left[\begin{array}{c|c} 0 & 0 \\ \mathbf{v} - \mathbf{a}_{11} \hat{\mathbf{y}}^{(1)} & B - \hat{\mathbf{y}}^{(1)} \mathbf{w}^T - \hat{L}_1 \hat{U}_1 \end{array} \right] \right| = \left| \left[\begin{array}{c|c} 0 & 0 \\ \mathbf{v} - \mathbf{a}_{11} \hat{\mathbf{y}}^{(1)} & A_1 - \hat{L}_1 \hat{U}_1 \end{array} \right] \right| \\ &\leq \left| \left[\begin{array}{c|c} 0 & 0 \\ \mathbf{a}_{11} (\hat{\mathbf{y}}^{(1)} - \mathbf{y}^{(1)}) & A_1 - \hat{A}_1 \end{array} \right] \right| + \left| \left[\begin{array}{c|c} 0 & 0 \\ \mathbf{v} - \mathbf{a}_{11} \mathbf{y}^{(1)} & \hat{A}_1 - \hat{L}_1 \hat{U}_1 \end{array} \right] \right| \end{aligned}$$

$$\begin{aligned}
&\leq \left| \left[\frac{0}{a_{11} \hat{u} |v| / |a_{11}|} \mid \frac{0}{A_1 - \hat{A}_1} \right] \right| + \left| \left[\frac{0}{0} \mid \frac{0}{\hat{A}_1 - \hat{L}_1 \hat{U}_1} \right] \right| \\
&\leq 3\hat{u} \left[\frac{0}{|v|} \mid \frac{0}{|A_1|} \right] + 3n\hat{u} \left[\frac{0}{0} \mid \frac{0}{|A_1| + |\hat{L}_1| |\hat{U}_1|} \right] \\
&\leq 3(n+1)\hat{u} \left[\frac{|a_{11}|}{|v|} \mid \frac{|w^T|}{|A_1| + |\hat{L}_1| |\hat{U}_1|} \right] = 3(n+1)\hat{u} (|A| + |\hat{L}| |\hat{U}|).
\end{aligned}$$

Und das war's dann auch schon. □

Beweis von Satz 4.21: Aus Proposition A.1 erhalten wir, daß

$$(\hat{L} + F) \hat{y} = b, \quad (\hat{U} + E) \hat{x} = \hat{y},$$

wobei

$$|E| \leq n\hat{u}|\hat{U}| + O(\hat{u}^2), \quad |F| \leq n\hat{u}|\hat{L}| + O(\hat{u}^2),$$

also

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

wobei

$$|G| = |\hat{L}E + F\hat{U} + FE| \leq 2n\hat{u} |\hat{L}| |\hat{U}| + O(\hat{u}^2).$$

Nach Proposition A.3 ist damit $\hat{A}\hat{x} = b$, wobei

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

□

A.2 Nachiteration

Es ist ganz interessant, sich klarzumachen, wie und warum die Nachiteration funktioniert, und vor allem, daß das nicht so ganz trivial ist.

Nehmen wir an, wir berechnen die arithmetischen Operationen in (4.38) mit einer Genauigkeit

$$\tilde{u} \leq \hat{u}.$$

Außerdem machen wir eine Annahme an die Stabilität unseres Lösungsverfahrens, nämlich, daß für jede rechte Seite b die berechnete Lösung \hat{x} die Beziehung

$$(A + E) \hat{x} = b, \quad \|A^{-1}E\|_{\infty} < 1, \quad (\text{A.8})$$

erfüllt; siehe Bedingung (3.9) in Satz 3.9.

In jeder Zeile von $A \otimes \hat{x}_{j-1}$ berechnen wir ein Skalarprodukt auf \mathbb{R}^n und zusammen mit der Subtraktion erhalten wir also, daß die berechnete Größe \tilde{r}_j den Wert

$$|\tilde{r}_j - \mathbf{b} + A\hat{x}_{j-1}| \leq (n+1) \tilde{u} (|\mathbf{b}| + |A| |\hat{x}_{j-1}|) + O(\tilde{u}^2). \quad (\text{A.9})$$

Dann rundet man diesen Wert auf „Arbeitsgenauigkeit“ \hat{u} , also ist

$$|\hat{r}_j - \tilde{r}_j| \leq \hat{u} |\tilde{r}_j|$$

Also:

$$\begin{aligned} |\hat{r}_j - \mathbf{b} + A\hat{x}_{j-1}| &\leq |\tilde{r}_j - \mathbf{b} + A\hat{x}_{j-1}| + |\hat{r}_j - \tilde{r}_j| \\ &\leq |\tilde{r}_j - \mathbf{b} + A\hat{x}_{j-1}| + \hat{u} |\tilde{r}_j| \\ &\leq |\tilde{r}_j - \mathbf{b} + A\hat{x}_{j-1}| + \hat{u} (|\mathbf{b} - A\hat{x}_{j-1}| + |\tilde{r}_j - \mathbf{b} + A\hat{x}_{j-1}|) \\ &= \hat{u} |\mathbf{b} - A\hat{x}_{j-1}| + (1 + \hat{u}) |\tilde{r}_j - \mathbf{b} + A\hat{x}_{j-1}| \end{aligned}$$

und mit (A.9) ergibt sich

$$|\hat{r}_j - \mathbf{b} + A\hat{x}_{j-1}| \leq \hat{u} |\mathbf{b} - A\hat{x}_{j-1}| + (1 + \hat{u}) (n+1) \tilde{u} (|\mathbf{b}| + |A| |\hat{x}_{j-1}|) + O(\tilde{u}^2). \quad (\text{A.10})$$

Mit $f_j := \hat{r}_j - \mathbf{b} + A\hat{x}_{j-1}$ ergibt sich dann

$$\begin{aligned} |f_j| &\leq \hat{u} |A(\hat{x}_{j-1} - \mathbf{x})| \\ &\quad + (1 + \hat{u}) (n+1) \tilde{u} (|A||\mathbf{x}| + |A||\mathbf{x}| + |A| |\hat{x}_{j-1} - \mathbf{x}|) + O(\tilde{u}^2) \\ &\leq (\hat{u} + (1 + \hat{u})(n+1)\tilde{u}) |A| |\hat{x}_{j-1} - \mathbf{x}| + 2(1 + \hat{u}) (n+1)\tilde{u} |A||\mathbf{x}| + O(\tilde{u}^2) \\ &= (\hat{u} + (n+1)\tilde{u}) |A| |\hat{x}_{j-1} - \mathbf{x}| + 2(n+1)\tilde{u} |A||\mathbf{x}| + O(\hat{u} \tilde{u}). \quad (\text{A.11}) \end{aligned}$$

Die berechnete Korrekturgröße \hat{y}_j erfüllt dann

$$(A + E_j) \hat{y}_j = \hat{r}_j,$$

wobei¹⁹⁵

$$(A + E_j)^{-1} = (I + A^{-1}E_j)^{-1} A^{-1} =: (I + F_j) A^{-1}, \quad \|F_j\|_\infty \leq \frac{\|A^{-1}E_j\|_\infty}{1 - \|A^{-1}E_j\|_\infty}. \quad (\text{A.12})$$

Deswegen ist

$$\begin{aligned} \hat{y}_j &= (A + E_j)^{-1} \hat{r}_j = (I + F_j) A^{-1} \hat{r}_j \\ &= (I + F_j) A^{-1} (\mathbf{e}_j + \mathbf{b} - A\hat{x}_{j-1}) = (I + F_j) (A^{-1} \mathbf{e}_j + \mathbf{x} - \hat{x}_{j-1}), \quad (\text{A.13}) \end{aligned}$$

¹⁹⁵Der inzwischen wohlbekannte Trick mit der geometrischen Reihe ...

und

$$|\hat{x}_j - (\hat{x}_{j-1} + \hat{y}_j)| \leq \hat{u} |\hat{x}_{j-1} + \hat{y}_j| \leq \hat{u} (|x - \hat{x}_{j-1}| + |x| + |\hat{y}_j|),$$

also, mit (A.13),

$$|\hat{x}_j - (\hat{x}_{j-1} + \hat{y}_j)| \leq \hat{u} (|x - \hat{x}_{j-1}| + |x|) + \hat{u} (I + |F_j|) \left(|A^{-1}| |f_j| + |x - \hat{x}_{j-1}| \right). \quad (\text{A.14})$$

Weiter geht's mit

$$\begin{aligned} \hat{x}_j - x &= \hat{x}_{j-1} + \hat{y}_j + \underbrace{(\hat{x}_j - (\hat{x}_{j-1} + \hat{y}_j))}_{=: z_j, \text{ (Rundung!)}} - x \\ &= \hat{x}_{j-1} - x + (I + F_j) \left(A^{-1} f_j + x - \hat{x}_{j-1} \right) + z_j \\ &= F_j (x - \hat{x}_{j-1}) + (I + F_j) A^{-1} f_j + z_j, \end{aligned}$$

und daher ist, unter Verwendung von (A.14) und (A.11)

$$\begin{aligned} |\hat{x}_j - x| &\leq |F_j| |x - \hat{x}_{j-1}| + (I + |F_j|) |A^{-1}| |f_j| + |z_j| \\ &\leq |F_j| |x - \hat{x}_{j-1}| + (I + |F_j|) |A^{-1}| |f_j| \\ &\quad + \hat{u} (|x - \hat{x}_{j-1}| + |x|) + \hat{u} (I + |F_j|) \left(|A^{-1}| |f_j| + |x - \hat{x}_{j-1}| \right) \\ &= ((1 + \hat{u}) |F_j| + 2\hat{u}I) |x - \hat{x}_{j-1}| + \hat{u} |x| + (1 + \hat{u}) (I + |F_j|) |A^{-1}| |f_j| \\ &\leq ((1 + \hat{u}) |F_j| + 2\hat{u}I) |x - \hat{x}_{j-1}| + \hat{u} |x| \\ &\quad + (1 + \hat{u}) (I + |F_j|) ((\hat{u} + (n+1)\tilde{u}) |A| |\hat{x}_{j-1} - x| + 2(n+1)\tilde{u} |A| |x|) \\ &\quad + O(\hat{u} \tilde{u}) \\ &= \left((1 + \hat{u}) |F_j| + (\hat{u} + (n+1)\tilde{u}) (I + |F_j|) |A^{-1}| |A| + O(\hat{u}^2) \right) |\hat{x}_{j-1} - x| \\ &\quad + (\hat{u}I + 2(n+1)\tilde{u} |A|) |x| + O(\hat{u} \tilde{u}) \end{aligned}$$

Somit ist

$$\begin{aligned} \|\hat{x}_j - x\|_\infty &\leq ((1 + \hat{u}) \|F_j\|_\infty + (n+2)\hat{u}(1 + \|F_j\|_\infty) \kappa_\infty(A)) \|\hat{x}_{j-1} - x\|_\infty \\ &\quad + (\hat{u} + 2(n+1)\tilde{u} \|A\|_\infty) \|x\|_\infty + O(\hat{u}^2) \\ &=: \gamma_j \|\hat{x}_{j-1} - x\|_\infty + \eta_j \|x\|_\infty + O(\hat{u}^2), \end{aligned} \quad (\text{A.15})$$

wobei

$$\gamma_j \leq \frac{(1 + \hat{u}) \|A^{-1} E_j\|_\infty + (n+2)\hat{u} \kappa_\infty(A)}{1 - \|A^{-1} E_j\|_\infty}. \quad (\text{A.16})$$

Unter der „Stabilitätsannahme“ $\|E_j\|_\infty \leq Cn\hat{u}\|A\|_\infty$ ist also $\gamma_j < 1$, falls

$$2(C+1)n\kappa_\infty(A) \ll \hat{u}^{-1} \quad (\text{A.17})$$

A.3 Der Satz von Weierstrass – Beweis

Eigentlich immer noch ein sehr schöner Beweis und fast der Originalbeweis von Bernstein aus [3].

Beweis von Satz 8.3: Ohne Einschränkung können wir annehmen, daß $I = [0, 1]$. Dann können wir eine Folge von approximierenden Polynomen *explizit* angeben, nämlich die **Bernsteinpolynome**¹⁹⁶

$$(B_n f)(x) = \sum_{j=0}^n f(j/n) \underbrace{\binom{n}{j} x^j (1-x)^{n-j}}_{=: B_j^n(x)} \in \Pi_n. \quad (\text{A.18})$$

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

$$\begin{aligned} \sum_{j=0}^n \frac{j(j-1)}{n^2} B_j^n(x) &= \sum_{j=0}^n \frac{j(j-1)}{n^2} \frac{n!}{j!(n-j)!} x^j (1-x)^{n-j} \\ &= \frac{n-1}{n} \sum_{j=2}^n \frac{(n-2)!}{(j-2)!(n-j)!} x^j (1-x)^{n-j} = \frac{n-1}{n} x^2 \sum_{j=0}^{n-2} B_j^{n-2}(x) \\ &= \frac{n-1}{n} x^2. \end{aligned}$$

Mit diesen drei Identitäten erhalten wir, daß für jedes $x \in [0, 1]$

$$\begin{aligned} \sum_{j=0}^n \left(\frac{j}{n} - x \right)^2 B_j^n(x) &= \sum_{j=0}^n \left(\frac{j^2}{n^2} - 2\frac{j}{n}x + x^2 \right) B_j^n(x) \\ &= \sum_{j=0}^n \left(\frac{j(j-1)}{n^2} + \frac{j}{n^2} - 2\frac{j}{n}x + x^2 \right) B_j^n(x) \\ &= \frac{n-1}{n} x^2 + \frac{1}{n} x - 2x^2 + x^2 = \frac{1}{n} x(1-x) \end{aligned}$$

¹⁹⁶Der Name kommt daher, daß der Beweis im Original von S. Bernstein [3] stammt, dem übrigens die wahrscheinlichkeitstheoretische Idee durchaus klar war.

Diese Formel erlaubt es uns nun zu zeigen, daß die Basispolynome recht gut lokalisiert sind: für $\delta > 0$ und $x \in [0, 1]$ ist

$$\sum_{\left| \frac{j}{n} - x \right| \geq \delta} B_j^n(x) \leq \frac{1}{\delta^2} \sum_{j=0}^n \left(\frac{j}{n} - x \right)^2 B_j^n(x) = \frac{x(1-x)}{n\delta^2} \leq \frac{1}{4n\delta^2}. \quad (\text{A.19})$$

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

$$|x - y| < \delta \quad \Rightarrow \quad |f(x) - f(y)| < \frac{\varepsilon}{2}.$$

Sei also $\varepsilon > 0$ vorgegeben und $\delta > 0$ passend gewählt. Dann ist, für beliebiges $x \in [0, 1]$,

$$\begin{aligned} |f(x) - B_n f(x)| &= \left| \sum_{j=0}^n (f(x) - f(j/n)) B_j^n(x) \right| \leq \sum_{j=0}^n |f(x) - f(j/n)| B_j^n(x) \\ &= \sum_{\left| \frac{j}{n} - x \right| < \delta} \underbrace{|f(x) - f(j/n)|}_{\leq \varepsilon/2} B_j^n(x) + \sum_{\left| \frac{j}{n} - x \right| \geq \delta} \underbrace{|f(x) - f(j/n)|}_{\leq 2\|f\|_I} B_j^n(x) \\ &\leq \frac{\varepsilon}{2} \sum_{j=0}^n B_j^n(x) + \frac{\|f\|_I}{2n\delta^2} = \frac{\varepsilon}{2} + \frac{\|f\|_I}{2n\delta^2}. \end{aligned}$$

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.

A.4 Schoenberg–Whitney und die totale Nichtnegativität der Kollokationsmatrix

Definition A.4 Sei $d = (d_0, \dots, d_n) \in \mathbb{R}^{n+1}$.¹⁹⁷ Eine Teilmenge $\Sigma = \{\sigma_1, \dots, \sigma_k\} \subset \{0, \dots, n\}$ heißt **Signatur** bezüglich d , falls

$$d_{\sigma_l} d_{\sigma_{l+1}} < 0, \quad l = 0, \dots, k-1. \quad (\text{A.20})$$

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

$$S(d) = \max_{\Sigma \in \Sigma(d)} \#\Sigma.$$

¹⁹⁷Wir betrachten jetzt also skalare Splinekurven, den Fall $d = 1$.

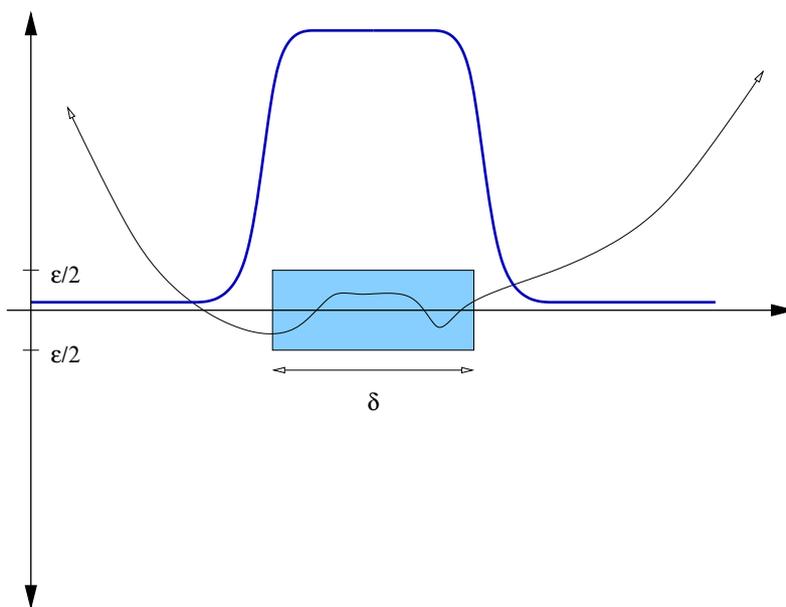


Abbildung A.1: Idee des Beweises von Satz 8.3: Auf dem farbig unterlegten δ -Intervall um den Punkt x weichen die Funktionen um maximal ε voneinander ab, und diese Abweichung ist wegen der gleichmäßigen Stetigkeit unabhängig von der Lage des δ -„Fensters“, auf dem Rest des Intervalls hat die (blaue) „Maskierungsfunktion“ hingegen so einen kleinen Wert, daß der Fehler dort machen kann, was er will.

Wir zeigen zuerst, daß die Zahl der Vorzeichenwechsel durch Knoteneinfügen nicht vergrößert wird.

Lemma A.5 *Ist T^* eine Verfeinerung von $T = T_{n,m}$ und ist $d^* \in \mathbb{R}^{1 \times n^*}$ der durch Knoteneinfügen aus d entstandene 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

$$t_0 \leq \dots \leq t_j \leq t^* \leq t_{j+1} \leq \dots \leq t_{n+m+1}$$

und $T^* = T \cup \{t^*\}$. Nach Satz 9.24 ergeben sich die Koeffizienten als

$$d_k^* = \begin{cases} d_k & k = 0, \dots, j - m, \\ u_0(t^* | I_k^m) d_{k-1} + u_1(t^* | I_k^m) d_k & k = j - m + 1, \dots, j, \\ d_{k-1} & k = j + 1, \dots, n + 1. \end{cases}$$

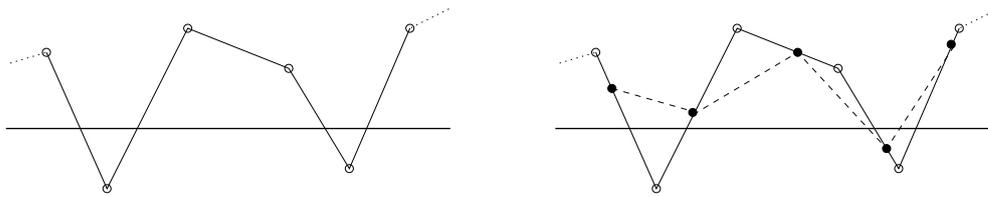


Abbildung A.2: Die Beweisidee von Lemma A.5. Die Variation des gestrichelten Polygons auf der rechten Seite, also $S(d^*)$, ist kleiner als die Variation des durchgezogenen Polygons (es werden ja zusätzliche Ecken dazwischengeschoben) mit schwarzen und weißen Ecken, bei dem man wiederum die schwarzen „Ecken“ vergessen kann, weil sie auf Verbindungsgeraden liegen.

Sei

$$d' = (d_0, \dots, d_{j-m}, d_{j-m+1}^*, d_{j-m+1}, d_{j-m+2}^*, \dots, d_{j-1}, d_j^*, d_j, d_{j+1}, \dots, d_n).$$

Da d_k^* eine **Konvexkombination** von d_{k-1} und d_k ist, $k = j - m + 1, \dots, j$, ist

$$S(d_{k-1}, d_k^*, d_k) = S(d_{k-1}, d_k),$$

also $S(d') = S(d)$. Andererseits ist

$$d^* = (d_0, \dots, d_{j-m}, d_{j-m+1}^*, \dots, d_j^*, d_j, d_{j+1}, \dots, d_n)$$

eine Teilfolge von d' und damit ist $S(d^*) \leq S(d') = S(d)$. □

Definition A.6 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 $f(y) \neq 0$.

Bemerkung A.7 Isolierte Nullstellen müssen nicht einfach sein: $f(x) = x^{2n}$ hat eine $2n$ -fache isolierte Nullstelle an $x = 0$. Umgekehrt hat die abgebrochene Potenz¹⁹⁸ $(\cdot - x_0)_+^n$ zwar jede Menge Nullstellen, nämlich $(-\infty, x_0]$, aber keine davon ist isoliert.

Lemma A.8 Sei $k \geq 0$, $T = T_{m,n}$ eine Knotenfolge und $d \in \mathbb{R}^{n+1}$ ein Kontrollvektor so daß $d_l = 0$, $l \notin \{j, \dots, j+k\}$ 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 $N_{m,T}d|_J \in \Pi_m$, $J = [t_\ell, t_{\ell+1}]$, $\ell = j, \dots, j+k+m$, 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!

¹⁹⁸Bei de Boor: „Stutzfunktion“ – klingt auch nicht besser.

Nach Lemma 9.14 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

$$\#\{x \in I^\circ : N_{m,T}d(x) = 0\} = \#\{x \in I^\circ : N_{m,T^*}d^*(x) = 0\} \leq S(d^*) \leq S(d) \leq k.$$

Außerdem kann eine isolierte Nullstelle durch Knoteneinfügen offensichtlich nur dann entstehen, wenn vorher ein Vorzeichenwechsel da war. \square

Beweis von Satz 9.6, „ \Leftarrow “: Nehmen wir an, es gäbe einen Vektor $0 \neq d \in \mathbb{R}^{n+1}$, so daß $N_m(\mathcal{X}|T)d = 0$, das heißt, daß $N_{m,T}(x_j|T) = 0$, $j = 0, \dots, n$. Ist nun für ein $j \in \{m, \dots, n\}$

$$0 = N_{m,T}d|_{I_j} = \sum_{k=j-m}^j d_k N_k^m(\cdot|T),$$

dann ist nach dem Algorithmus von de Boor

$$d_{j-m} = \dots = d_j = 0$$

und $N_{m,T}d$ zerfällt in

$$N_{m,T}d = \underbrace{\sum_{k=0}^{j-m-1} d_k N_k^m(\cdot|T)}_{=:f_1} + \underbrace{\sum_{k=j+1}^n d_k N_k^m(\cdot|T)}_{=:f_2},$$

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 $N_m(\mathcal{X}|T)d = 0$, und es j und k gibt, so daß $d_l = 0$, $l \notin \{j, \dots, j+k\}$ 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, \dots, x_{j+k} , also müßte $N_{m,T}d$ dort $k+1$ isolierte Nullstellen haben, was aber Lemma A.8 widerspricht. Also ist die Matrix $N_m(\mathcal{X}|T)$ nichtsingulär. \square

Beweis von Satz 9.6, 2: Sei t_j ein Knoten der Vielfachheit $< m$ und $t_j \leq t^* \leq t_{j+1}$. Nach Satz 9.24 ist dann für die durch Einfügen von t^* entstandene Knotenfolge T^*

$$N_k^m(\cdot|T) = \sum_{\ell=0}^{n+1} \alpha_{k,\ell} N_\ell^m(\cdot|T^*), \quad (\text{A.21})$$

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 gilt¹⁹⁹, daß

$$\alpha_{k,\ell} \neq 0 \quad \Rightarrow \quad \ell \in \{k, k+1\}. \quad (\text{A.22})$$

Wir setzen

$$\mathcal{X}_I = \{x_i : i \in I\}, \quad I \subseteq \{0, \dots, n\},$$

und definieren die *Spaltenvektoren*

$$N_j^m(\mathcal{X}_I|T) := [N_j^m(x_i|T) : i \in I], \quad j = 0, \dots, n, \quad I \subseteq \{0, \dots, n\}.$$

Dann ist, für $I, J \subset \{0, \dots, n\}$, $\#I = \#J = k$,²⁰⁰

$$\begin{aligned} \det N_m(\mathcal{X}|T)(I, J) &= \det [N_j^m(\mathcal{X}_I|T) : j \in J] \\ &= \det [N_{j_1}^m(\mathcal{X}_I|T), N_j^m(\mathcal{X}_I|T) : j \in J \setminus \{j_1\}] \\ &= \det \left[\sum_{\ell_1=0}^{n+1} \alpha_{j_1, \ell_1} N_{\ell_1}^m(\mathcal{X}_I|T^*), N_j^m(\mathcal{X}_I|T) : j \in J \setminus \{j_1\} \right] \\ &= \sum_{\ell_1=0}^{n+1} \alpha_{j_1, \ell_1} \det [N_{\ell_1}^m(\mathcal{X}_I|T^*), N_j^m(\mathcal{X}_I|T) : j \in J \setminus \{j_1\}] \\ &\quad \vdots \\ &= \sum_{\ell_1=0}^{n+1} \alpha_{j_1, \ell_1} \cdots \sum_{\ell_k=0}^{n+1} \alpha_{j_k, \ell_k} \det [N_r^m(\mathcal{X}_I|T^*) : r = \ell_1, \dots, \ell_k] \\ &= \sum_{\ell_1=0}^{n+1} \cdots \sum_{\ell_k=0}^{n+1} \alpha_{j_1, \ell_1} \cdots \alpha_{j_k, \ell_k} \det N_m(\mathcal{X}|T^*)(I, \{\ell_1, \dots, \ell_k\}). \end{aligned}$$

Da, nach (A.22), $\alpha_{j_r, \ell_r} = 0$ falls $\ell_r \notin \{j_r, j_r+1\}$, und da²⁰¹ $j_1 < \dots < j_k$, müssen die Elemente jeder Indexmenge $L = \{\ell_1, \dots, \ell_k\}$ in der obigen Summe auf alle Fälle die Bedingung $\ell_1 \leq \dots \leq \ell_k$ erfüllen. Tritt aber irgendwo Gleichheit ein, d.h. ist $\ell_r = \ell_{r+1}$, dann enthält die Matrix $N_m(\mathcal{X}|T^*)(I, L)$ zwei *identische* Spalten und damit ist die

¹⁹⁹Wegen der beteiligten $\delta_{k,\ell}$ und $\delta_{k,\ell-1}$.

²⁰⁰Zur Erinnerung: $N_m(\mathcal{X}|T)$ 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.

²⁰¹Dies ist die kanonische Annahme, daß J *aufsteigend* numeriert ist

Determinante gleich Null. Damit läuft die Summe über alle $\ell_1 < \dots < \ell_k$, also über alle $L \subseteq \{0, \dots, n+1\}$ ²⁰² mit $\#L = k$. Mit anderen Worten, für

$$\alpha(J, L) := \prod_{r=1}^k \alpha_{j_r, \ell_r} \geq 0, \quad J \subseteq \{0, \dots, n\}, \quad L \subseteq \{0, \dots, n+1\}, \quad \#J = \#L,$$

haben wir, daß

$$\det N_m(\mathcal{X}|T)(I, J) = \sum_{\#L=k} \alpha(J, L) \det N_m(\mathcal{X}|T^*)(I, L). \quad (\text{A.23})$$

Fügen wir nun noch einen Knoten ein, dann ergibt sich entsprechend für die neue Knotenfolge T^{**}

$$\begin{aligned} \det N_m(\mathcal{X}|T)(I, J) &= \sum_{\#L=k} \alpha(J, L) \det N_m(\mathcal{X}|T^*)(I, L) \\ &= \sum_{\#L=k} \alpha(J, L) \sum_{\#L'=k} \alpha(L, L') \det N_m(\mathcal{X}|T^{**})(I, L') \\ &= \sum_{\#L'=k} \underbrace{\sum_{\#L=k} \alpha(J, L) \alpha(L, L')}_{=:\alpha^2(J, L')} \det N_m(\mathcal{X}|T^{**})(I, L'). \end{aligned}$$

In diesem Sinne haben wir, für beliebiges $N \in \mathbb{N}$, nach Einfügen von N Knoten²⁰³, für das resultierende T^{N*} ,

$$\det N_m(\mathcal{X}|T)(I, J) = \sum_{\#L=k} \alpha^N(J, L) \det N_m(\mathcal{X}|T^{N*})(I, L), \quad (\text{A.24})$$

wobei

$$\alpha^N(J, L) = \sum_{\#L_1=k} \dots \sum_{\#L_{N-1}=k} \prod_{r=1}^{N+1} \alpha(L_r, L_{r+1}), \quad L_r \subset \{0, \dots, n+r\}, \quad r = 1, \dots, N. \quad (\text{A.25})$$

Wir wählen nun T^{N*} ²⁰⁴ so dicht, daß zwischen je zwei Interpolationspunkten x_j und x_{j+1} mindestens $m+1$ Knoten liegen und daher, für $k = 0, \dots, n$,

$$N_j^m(x_k|T^*) \neq 0 \quad \Rightarrow \quad N_j^m(x_\ell|T^*) = 0, \quad \ell \neq k.$$

Also hat *jede* Spalte *jeder* der (oberen Dreiecks-) Matrizen $N_m(\mathcal{X}|T^{N*})(I, L)$, $L \subset \{0, \dots, n+N\}$, höchstens einen von Null verschiedenen Eintrag. Liegt wenigstens einer davon

²⁰²Achtung: Der Index kann jetzt auch $n+1$ werden!

²⁰³Es genügt, wenn diese Knoten einfach, also $t_1^* < t_2^* < \dots < t_N^*$ und neu, also $t_j^* \notin T$, sind.

²⁰⁴Genauer, die eingefügten Knoten.

nicht auf der Diagonale, dann ist $\det N_m(\mathcal{X}|T^{N^*})(I, L) = 0$, andernfalls ergibt sich

$$\det N_m(\mathcal{X}|T^{N^*})(I, L) = \prod_{r=1}^k \underbrace{N_{\ell_r}^m(x_{i_r}|T^*)}_{>0} > 0.$$

Setzt man das in (A.24) und berücksichtigt man (A.25), dann heißt das gerade, daß $\det N_m(\mathcal{X})(I, J) > 0$. \square

A.5 Eine Anwendung nichtlinearer Gleichungen: 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 Getränkeboxen, genauer Stapel von Getränkeboxen²⁰⁵, bewegt und zwar über ein Hindernis hinweg. Um dieses Hindernis zu umfahren, muss die Bahn des Roboters durch einen bestimmten Punkt hindurchgeführt werden. Aus technischen Gründen²⁰⁶ ist die Bahn eine Kombination aus Geradenstücken und Kreisbögen mit *einem* festen Radius r . Damit die Bierkästen nicht umfallen

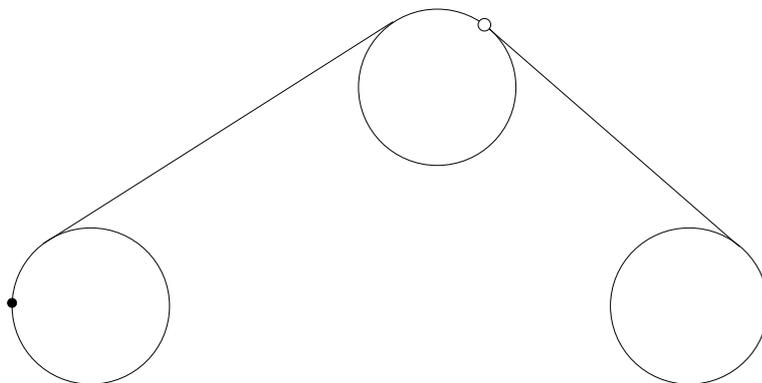


Abbildung A.3: Eine zulässige Bahn durch den vorgegebenen Punkt. Die Kreise mit dem vorgegebenen Radius sind eingezeichnet.

muss die Bahn ausserdem im Anfangs- und Endpunkt eine senkrechte Tangente haben, siehe Abb. A.3. Die Aufgabe ist nun:

Wie bestimmt man den kürzesten derartigen Weg.

²⁰⁵Daher der interne „Codename“ des Projekts: *Das Bierkastenproblem*

²⁰⁶So ist das Ding halt einfach gebaut.

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. A.4. Die beiden Mittelpunkte sind durch Anfangs- bzw.

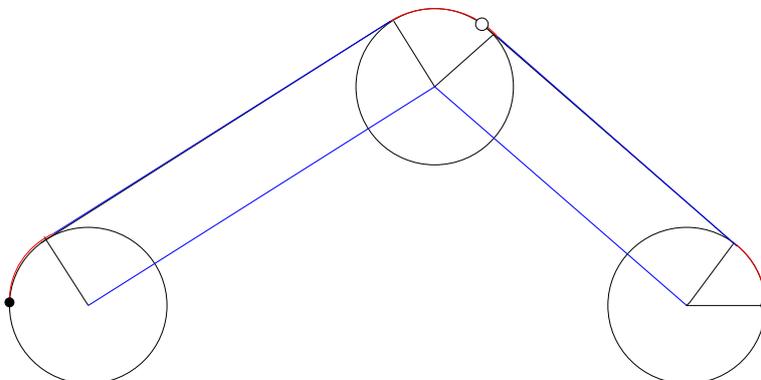


Abbildung A.4: Die Kreis- und Geradenanteile. Genau genommen interessiert einen also nur ein Pfad vom ersten Kreismittelpunkt über einem Punkt, dessen Abstand zum „Ausweichpunkt“ mit dem Radius übereinstimmt zum Mittelpunkt des zweiten Kreises.

Endpunkt und Radius eindeutig 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. A.5,

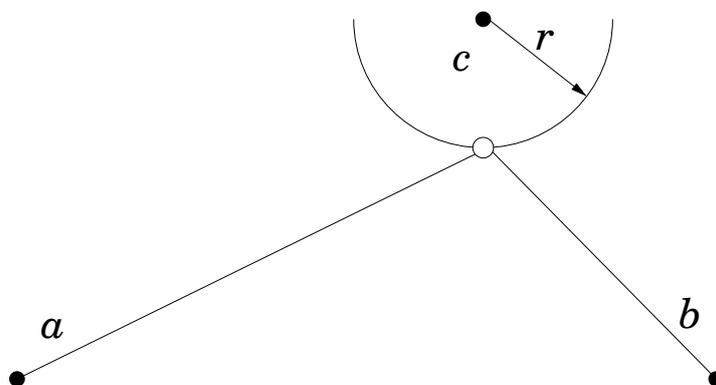


Abbildung A.5: Das wirkliche Optimierungsproblem.

und zwar so, daß der **Gesamtweg** minimal wird. Das Optimierungsproblem besteht also darin, einen Punkt x auf dem Kreis so zu wählen, daß der Gesamtweg minimiert wird, also

$$\min_x w(x) := \min_x \|x - a\|_2 + \|x - b\|_2, \quad \|x - c\|_2 = r. \quad (\text{A.26})$$

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

$$\left\| \frac{x + x'}{2} \right\|_2 < \frac{1}{2} (\|x\|_2 + \|x'\|_2)$$

ist. Wären nun x, x' zwei Punkte auf dem Kreis, für die der Weg $w(x) = w(x')$ minimal ist, dann ist

$$\begin{aligned} w\left(\frac{x + x'}{2}\right) &= \left\| \frac{x + x'}{2} - a \right\|_2 + \left\| \frac{x + x'}{2} - b \right\|_2 \\ &= \left\| \frac{1}{2} [(x - a) + (x' - a)] \right\|_2 + \left\| \frac{1}{2} [(x - b) + (x' - b)] \right\|_2 \\ &< \frac{1}{2} (\|x - a\|_2 + \|x' - a\|_2 + \|x - b\|_2 + \|x' - b\|_2) = \frac{1}{2} (w(x) + w(x')), \end{aligned}$$

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 $\frac{1}{2}(x + x')$ nicht zutrifft. Mit anderen Worten: $w(x)$ fällt monoton bis $w(x^*)$ und steigt von dort aus wieder.

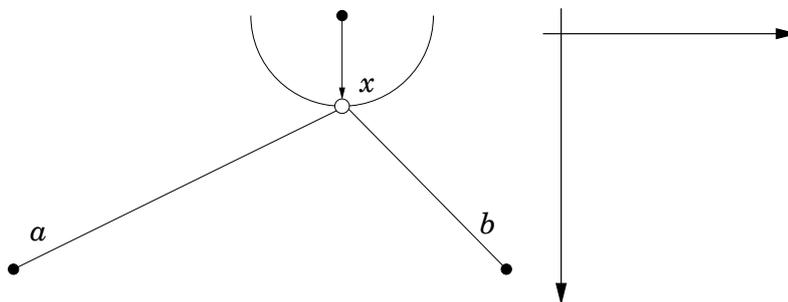


Abbildung A.6: Parametrisierung und Koordinatensystem.

Nun parametrisieren wir x als

$$x = c + r \begin{bmatrix} \sin \phi \\ \cos \phi \end{bmatrix}, \quad \phi \in [-\pi, \pi],$$

nehmen der Einfachheit an, daß $r = 1$ ist²⁰⁷, 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

$$w'(\phi) = \frac{(c - a)^T s'(\phi)}{\ell_a(\phi)} + \frac{(c - b)^T s'(\phi)}{\ell_b(\phi)}, \quad s(\phi) = \begin{bmatrix} \sin \phi \\ \cos \phi \end{bmatrix}.$$

Auf der anderen Seite ist aber auch

$$\begin{aligned} \ell_a^2(\phi) &= (c_1 + \sin \phi - a_1)^2 + (c_2 + \cos \phi - a_2)^2 \\ &= (c_1 - a_1)^2 + 2(c_1 - a_1) \sin \phi + \sin^2 \phi + (c_2 - a_2)^2 + 2(c_2 - a_2) \cos \phi + \cos^2 \phi \\ &= 1 + \|c - a\|_2^2 + 2(c - a)^T s(\phi) =: 1 + \|c - a\|_2^2 + 2f_a(\phi), \end{aligned}$$

also

$$w'(\phi) = \frac{f'_a(\phi)}{\sqrt{1 + \|c - a\|_2^2 + 2f_a(\phi)}} + \frac{f'_b(\phi)}{\sqrt{1 + \|c - b\|_2^2 + 2f_b(\phi)}}.$$

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

$$g'(\phi) = \ell'_b(\phi) f'_a(\phi) + \ell_b(\phi) f''_a(\phi) + \ell'_a(\phi) f'_b(\phi) + \ell_a(\phi) f''_b(\phi)$$

²⁰⁷So viel Normalisierung kann und darf schon sein.

ist, können wir nun (versuchen,) ϕ über eine univariate Newton–Iteration oder über Bisektion zu bestimmen.

Betrachtet man die Menge der Punkte x , für die $w(x)$ 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 Suche nach dem kürzesten Weg ist äquivalent zum Auffinden der Berührellipse mit der großen Halbachse α , siehe Abb. A.7. Dazu

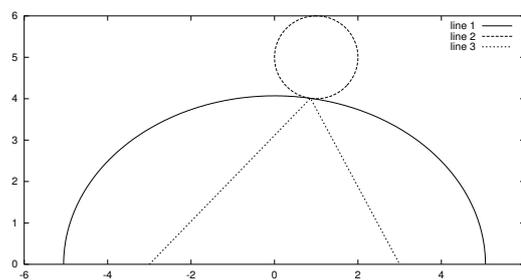


Abbildung A.7: Berührellipse an die Kugel und der zugehörige kürzeste Weg.

empfehl 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

$$\frac{x^2}{\alpha^2} + \frac{y^2}{\beta^2} = 1, \quad \alpha^2 - \beta^2 = e^2 = \|a - b\|_2^2, \quad (\text{A.27})$$

und die der Kugel ja bekanntlich

$$(x - c_1)^2 + (y - c_2)^2 = r^2. \quad (\text{A.28})$$

Dieses Gleichungssystem aus (A.27) und (A.28) hat im allgemeinen vier Lösungen, beispielsweise, wenn $c = 0$ und $\beta < r < \alpha$ ist. Zwei der Lösungen sind anschaulich sofort klar, nämlich der nächste Punkt wie in Abb. A.7 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 Hörtetest, siehe [34].

Um das Problem ausgehend von dieser geometrischen Überlegung numerisch anzugehen, verwenden wir die parametrischen Darstellungen von Kugel und Ellipse als

$$K(\phi) = c - r \begin{bmatrix} \sin \phi \\ \cos \phi \end{bmatrix}, \quad E(\psi) = \begin{bmatrix} \alpha \sin \psi \\ \beta \cos \psi \end{bmatrix}, \quad \phi, \psi \in [-\pi, \pi].$$

Wir müssen also α , ϕ und ψ so bestimmen, daß wir einen Berührungspunkt erhalten, daß also die Tangenten

$$K'(\phi) = -r \begin{bmatrix} \cos \phi \\ -\sin \phi \end{bmatrix}, \quad E'(\psi) = \begin{bmatrix} \alpha \cos \psi \\ -\beta \sin \psi \end{bmatrix}$$

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 Nebenbedingungen

$$\begin{aligned} 0 &= s_\phi^2 + c_\phi^2 - 1, \\ 0 &= s_\psi^2 + c_\psi^2 - 1, \\ 0 &= \alpha^2 - \beta^2 - e^2. \end{aligned}$$

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

$$y = c - \frac{r}{\|c\|_2} c =: c - r \begin{bmatrix} s_\phi \\ c_\phi \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} s_\phi \\ c_\phi \end{bmatrix} = \frac{c}{\|c\|_2}.$$

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

$$\tilde{s}_\psi = \frac{c_1 - r s_\phi}{\alpha}, \quad \tilde{c}_\psi = \frac{c_2 - r c_\phi}{\beta}$$

erhalten. Das können wir noch zu

$$s_\psi = \frac{\tilde{s}_\psi}{\tilde{s}_\psi^2 + \tilde{c}_\psi^2} \quad \text{und} \quad c_\psi = \frac{\tilde{c}_\psi}{\tilde{s}_\psi^2 + \tilde{c}_\psi^2}$$

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.

A.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 Horner Schema zur Auswertung eines Polynoms

$$p(x) = \sum_{j=0}^n p_j x^j, \quad p_n \neq 0, \quad x \in \mathbb{R},$$

an der Stelle $\xi \in \mathbb{R}$. Die Berechnungsregel lautet

$$q_n = p_n, \quad q_j = q_j(\xi) = \xi q_{j+1} + p_j, \quad j = n-1, \dots, 0 \quad (\text{A.29})$$

wobei dann $p(\xi) = q_0(\xi)$.

Lemma A.9 Für $\xi \in \mathbb{R}$ erfüllt das Polynom

$$q(x) := \sum_{j=0}^{n-1} q_{j+1}(\xi) x^j, \quad x \in \mathbb{R},$$

die Bedingung

$$p(x) = q_0(\xi) + (x - \xi)q(x), \quad x \in \mathbb{R}. \quad (\text{A.30})$$

Beweis: Iteration von (A.29) liefert, daß

$$q_j = \sum_{k=0}^{n-j} p_{j+k} \xi^k, \quad j = 0, \dots, n,$$

also ist

$$\begin{aligned} (x - \xi) q(x) + q_0 &= (x - \xi) \sum_{j=0}^{n-1} q_{j+1} x^j + q_0 = q_0 + \underbrace{\sum_{j=0}^{n-1} q_{j+1} x^{j+1}}_n - \sum_{j=0}^{n-1} \xi q_{j+1}(\xi) x^j, \\ &= \sum_{j=0}^n q_j x^j \end{aligned}$$

und

$$\begin{aligned} \sum_{j=0}^{n-1} \xi q_{j+1}(\xi) x^j &= \sum_{j=0}^{n-1} \sum_{k=0}^{n-j-1} p_{j+1+k} \xi^{k+1} x^j = \sum_{j=0}^{n-1} \sum_{k=1}^{n-j} p_{j+k} \xi^k x^j \\ &= \sum_{j=0}^n \underbrace{\sum_{k=0}^{n-j} p_{j+k} \xi^k}_{=q_j} x^j - \sum_{j=0}^n p_j x^j = \sum_{j=0}^n q_j x^j - p(x). \end{aligned}$$

□

Korollar A.10 Für $\xi \in \mathbb{R}$ ist

$$p'(\xi) = q(\xi). \quad (\text{A.31})$$

Beweis: Ableiten von (A.30) nach x ergibt

$$p'(x) = q(x) + (x - \xi) q'(x),$$

was mit $x = \xi$ sofort (A.31) liefert. □

Um q auszuwerten kann man nun wieder auf das Horner-schema zurückgreifen und die gerade berechneten Werte $q_j(\xi)$ verwenden. Das ergibt ein **zweizeiliges Horner-schema**

$$r_n = q_n = p_n, \quad q_j = \xi q_{j+1} + p_j, \quad r_j = \xi r_{j+1} + q_j, \quad j = n-1, \dots, 0. \quad (\text{A.32})$$

Korollar A.11 Für ein Polynom $p \in \Pi_n$ und $\xi \in \mathbb{R}$ berechnet das zweizeilige Horner-schema (A.32) die Werte $p(\xi) = q_0$ und $p'(\xi) = r_1$.

```

%#
%# 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 A.1 Horner2Z.m: Das zweizeilige Hornerschema.

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.

Proposition A.12 Sei $p \in \Pi_n$ ein Polynom mit n reellen Nullstellen $\xi_1 \leq \dots \leq \xi_n$. Dann konvergiert für alle Startwerte $x^{(0)} \geq \xi_n$ bzw. $x^{(0)} \leq \xi_1$ das Newton-Verfahren gegen ξ_n bzw. ξ_1 .

Beweis: Wir bemerken zuerst, daß nach dem Satz von Rolle alle Nullstellen von p', p'', \dots 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 an²⁰⁸. Da, für $x > \xi_n$,

$$0 < p(x) = \underbrace{p(\xi_n)}_{=0} + p'(\xi)(x - \xi_n), \quad \xi \in [\xi_n, x],$$

und da p' rechts von ξ_n keine Nullstelle mehr hat, ist also $p'(x) > 0, x > \xi_n$. Sei nun $\xi' \leq \xi_n$ die größte Nullstelle von p' , dann ist für $x > \xi_n \geq \xi'$ auch

$$0 < p'(x) = \underbrace{p'(\xi')}_{=0} + p''(\xi)(x - \xi'), \quad \xi \in [\xi', x],$$

²⁰⁸Das ist keine Einschränkung, ansonsten ersetzen wir p durch $-p$.

```
##
## 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 A.2 PolyNewton.m: Nullstellen von Polynomen mit dem Newton-Verfahren.

das heißt, $p''(x) > 0$, $x > \xi_n$, und mit Iteration dieses Arguments erhält man, daß

$$\operatorname{sgn} p^{(j)}(x) = \operatorname{sgn} p(x), \quad x > \xi_n, \quad j = 1, \dots, n.$$

Insbesondere ist p rechts von ξ_n konvex und damit ist

$$x^{(k+1)} = x^{(k)} - \underbrace{\frac{p(x^{(k)})}{p'(x^{(k)})}}_{>0} < x^{(k)},$$

solange $x^{(k)} > \xi_n$. Bleibt zu zeigen, daß auch $x^{(k+1)} > \xi_n$. Dazu bemerken wir, daß

$$\begin{aligned} 0 &= p(\xi_n) = p(x^{(k)}) + (\xi_n - x^{(k)}) p'(x^{(k)}) + \underbrace{\frac{(\xi_n - x^{(k)})^2}{2} p''(\xi)}_{>0} \\ &> p(x^{(k)}) + (\xi_n - x^{(k)}) p'(x^{(k)}) \end{aligned}$$

und, nach der Iterationsregel, $p(x^{(k)}) = (x^{(k)} - x^{(k+1)}) p'(x^{(k)})$, also

$$0 > (x^{(k)} - x^{(k+1)} + \xi_n - x^{(k)}) p'(x^{(k)}) = (\xi_n - x^{(k+1)}) \underbrace{p'(x^{(k)})}_{>0},$$

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

$$\left| \frac{p(x^{(k)})}{p'(x^{(k)})} \right| = |x^{(k+1)} - x^{(k)}| \rightarrow 0$$

konvergiert, muß $p(x^*) = 0$ und damit $x^* = \xi_n$ sein. \square

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 Nullstelle²⁰⁹, 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 gleichzeitig²¹⁰ als Startwert für das nächste Newton-Verfahren.

²⁰⁹Am besten natürlich die betragsgrößte oder betragskleinste.

²¹⁰In der Hoffnung, daß sie eine extremale Nullstelle war.

```
##
## 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 A.3 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 A.4 PolyZeros.m: Newton-Verfahren mit Adividieren.

Beispiel A.13 Leider ist die hier vorgestellte Variante des Newton–Verfahrens nicht übermäßig stabil, insbesondere das Abdividieren führt zu Problemen.

1. Betrachten wir das Polynom

$$p(x) = \prod_{j=0}^{13} (x - 2^{-j}), \quad x \in \mathbb{R},$$

siehe [41, S. 259–260]. Startet man das Newton–Verfahren mit der größten Nullstelle $x^{(0)} = 1$, dann erhält man die folgenden Nullstellen²¹¹

	$\varepsilon = 10^{-6}$		$\varepsilon = \hat{u}$	
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	→		0.062352	0.0023729
0.03125			0.039506	0.26421
0.015625			→	

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	→		0.058106	0.070300
0.03125			0.054892	0.75654
0.015625			→	

2. Ein anderes „Monster“ ist das schon klassische Wilkinson–Polynom

$$p(x) = (x - 1) \cdots (x - 20), \quad x \in \mathbb{R},$$

mit den ganz sauber getrennten Nullstellen $1, \dots, 20$. Dieses Polynom hat die Rundungsfehleranalyse sehr stark motiviert und beeinflusst, siehe [47]. Die Details sind in [44, 45] 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.

²¹¹ „→“ bedeutet, daß die Rechnung nicht mehr terminiert, also das Verfahren sich „aufhängt“.

A.7 Quadratur nach Art des Gaußes

In diesem "Bonus-Abschnitt" sehen wir uns noch an, wie Gauß die Knoten und Gewichte der Gauß-Quadratur für das Intervall $[0, 1]$ hergeleitet hat. Gauß selbst hat mit *formalen* Potenzreihen gerechnet²¹² und sich um Fragen der Konvergenz, die man sich zu dieser Zeit auch nicht unbedingt gestellt hat, gekümmert. Man kann aber zeigen, daß alle betrachteten Reihen tatsächlich auf einem passenden Bereich konvergieren – und nur da sieht man sie sich dann auch an. Auch bei der Darstellung der meisten Rechenschritte versuche ich hier, den "Gaußschen Geist" beizubehalten, also eine (freie) Übertragung der Originalarbeit [9]. Doch zuerst die Übungsaufgaben für das Selbststudium.

Übung A.3 [Kettenbrüche] Seien Zahlen $v_k, w_k, k \in \mathbb{N}_0$, gegeben. Dann heißt, für $n \in \mathbb{N}$,

$$C_n := \frac{v_1}{w_1} + \frac{v_2}{w_2} + \cdots + \frac{v_n}{w_n} = \frac{v_1}{w_1 + \frac{v_2}{w_2 + \frac{v_3}{w_3 + \cdots + \frac{v_n}{w_n}}}}, \quad C_0 := 0,$$

n-te Konvergente des Kettenbruchs

$$\frac{v_1}{w_1} + \frac{v_2}{w_2} + \cdots = \frac{v_1}{w_1 + \frac{v_2}{w_2 + \frac{v_3}{w_3 + \cdots}}}$$

Zeigen Sie:

1. Sei $C_n = V_n/W_n$. Dann erfüllen V_n und W_n die Rekursionsformeln

$$\begin{aligned} V_0 &= 0, \quad V_1 = v_1, & V_{n+1} &= w_{n+1}V_n + v_{n+1}V_{n-1}, \\ W_0 &= 1, \quad W_1 = w_1, & W_{n+1} &= w_{n+1}W_n + v_{n+1}W_{n-1}, \end{aligned} \quad n \in \mathbb{N}.$$

2. Für $n \in \mathbb{N}_0$ gilt

$$\frac{V_n}{W_n} - \frac{V_{n+1}}{W_{n+1}} = (-1)^{n+1} \frac{v_1 \cdots v_{n+1}}{W_n W_{n+1}},$$

und damit die (formale) Identität

$$\frac{v_1}{w_1} + \frac{v_2}{w_2} + \cdots = \sum_{k=0}^{\infty} (-1)^k \frac{v_1 \cdots v_{k+1}}{W_k W_{k+1}}.$$

²¹²Ja, wirklich gerechnet!

3. (*D. Bernoulli*) Für jede Folge von reellen Zahlen $c_n, n \in \mathbb{N}_0$, mit $c_{n+1} \neq c_n$ gibt es einen Kettenbruch

$$\frac{v_1|}{|w_1|} + \frac{v_2|}{|w_2|} + \frac{v_3|}{|w_3|} + \dots,$$

so daß

$$c_n = \frac{v_1|}{|w_1|} + \dots + \frac{v_n|}{|w_n|}$$

ist, und zwar

$$v_1 = c_1, \quad w_1 = 1,$$

$$v_n = \frac{c_{n-1} - c_n}{c_{n-1} - c_{n-2}}, \quad w_n = \frac{c_n - c_{n-2}}{c_{n-1} - c_{n-2}}, \quad n \geq 2.$$

◇

Übung A.4 [Quadratur á la Gauß] Seien paarweise verschiedene Knoten $x_0, x_1, \dots, x_n \in [0, 1]$ gegeben und sei

$$\omega(x) := (x - x_0) \cdots (x - x_n) \in \Pi_{n+1}.$$

Außerdem seien $\omega_1(x)$ und $\omega_2(x)$ gegeben durch

$$\omega(x) \sum_{j=0}^{\infty} \mu_j x^{-j} =: \omega_1(x) + \omega_2(x), \quad \omega_1 \in \Pi_n, \quad \omega_2(x) = \sum_{j=1}^{\infty} c_j x^{-j}.$$

1. Zeigen Sie:

$$\int_0^1 \frac{\omega(t)}{t - x_k} dt = \omega_1(x_k), \quad k = 0, \dots, n,$$

und bestimmen Sie daraus die Gewichte $w_j^*, j = 0, \dots, n$, der zugehörigen interpolatorischen Quadraturformel

$$Q_n^*(f) = \sum_{j=0}^n w_j^* f(x_j).$$

2. Seien $\theta_j, j \in \mathbb{N}_0$, die *Fehlermomente* der Quadraturformel Q_n^* , d.h.,

$$\theta_j := \int_0^1 t^j dt - Q_n^*((\cdot)^j),$$

und sei

$$\Theta(x) := \sum_{j=1}^{\infty} \theta_{j-1} x^{-j}.$$

Untersuchen Sie den Term

$$\omega(x) \sum_{j=0}^n \frac{w_j^*}{x - x_j}$$

und zeigen Sie, daß

$$\Theta(x) = \frac{\omega_2(x)}{\omega(x)}, \quad x \in \mathbb{R} \setminus \{0\}.$$

Hinweis: Verwenden Sie für $0 \neq t \in \mathbb{R}$ eine Taylorentwicklung der Funktion $(t - x)^{-1}$ nach x um den Punkt $x = 0$.

3. Betrachten Sie die Kettenbruchentwicklung

$$\sum_{j=1}^{\infty} \mu_{j-1} x^{-j} = \frac{1}{|x} - \frac{\alpha_1|}{|x} - \frac{\alpha_2|}{|x} - \dots, \quad \alpha_i > 0,$$

und zeigen Sie (Notation wie in der vorhergehenden Aufgabe): W_n ist ein Polynom vom Grad n in x und setzt man $\omega(x) = W_{n+1}(x)$, so gilt

$$\omega_2(x) = \sum_{i=2n+2}^{\infty} c_i x^{-i}.$$

4. Folgern Sie daraus, daß die Quadraturformel bezüglich der Nullstellen von $W_{n+1}(x)$ (warum hat dieses Polynom $n + 1$ Nullstellen?) Exaktheitsgrad $2n + 1$ hat.

◇

Bestimmung der Gewichte. Da wir eine interpolatorische Quadraturformel im Auge haben, ergeben sich die Gewichte als $w_j = I(\ell_j)$, $j = 0, \dots, n$. Bedenkt man, daß

$$\ell_j = \frac{\omega}{\omega'(x_j) (\cdot - x_j)}, \quad j = 0, \dots, n, \quad (\text{A.33})$$

dann besteht der "interessante" Teil der Bestimmung von w_j^* in der Berechnung des Wertes $I(\omega/(\cdot - x_j))$. Sei nun $\omega = x^{n+1} + a_n x^n + \dots + a_0$, dann ist, da $\omega(x_j) = 0$,

$$\omega(x) = \omega(x) - \omega(x_j) = (x^{n+1} - x_j^{n+1}) + a_n (x^n - x_j^n) + \dots + a_1 (x - x_j).$$

Da für $k = 0, \dots, n$

$$\frac{x^{k+1} - x_j^{k+1}}{x - x_j} = x^k + x^{k-1} x_j + \dots + x x_j^{k-1} + x_j^k,$$

können wir für $j = 0, \dots, n$ die Taylorentwicklung von $\frac{w_j^*}{x-x_j}$ nach x_j um $x_j = 0$ als

$$\frac{w_j^*}{x-x_j} = w_j^* \sum_{k=0}^{\infty} \frac{x_j^k}{k!} \left(\frac{d^k}{dx_j^k} \frac{1}{x-x_j} \right) \Big|_{x_j=0} = w_j^* \sum_{k=0}^{\infty} \frac{x_j^k}{k!} k! \frac{1}{(x-x_j)^{k+1}} \Big|_{x_j=0} = w_j^* \sum_{k=0}^{\infty} \frac{x_j^k}{x^{k+1}}$$

bestimmen und erhalten somit, daß

$$\begin{aligned} \sum_{j=0}^n \frac{w_j^*}{x-x_j} &= x^{-1} \underbrace{\sum_{j=0}^n w_j^*}_{=Q_n^*(1)} + x^{-2} \underbrace{\sum_{j=0}^n w_j^* x_j}_{=Q_n^*(\cdot)} + x^{-3} \underbrace{\sum_{j=0}^n w_j^* x_j^2}_{=Q_n^*(\cdot^2)} + \dots \\ &= \sum_{k=0}^{\infty} \frac{\mu_k - (\mu_k - Q_n^*(\cdot^k))}{x^{k+1}} = \sum_{k=1}^{\infty} (\mu_{k-1} - \theta_{k-1}) x^{-k} = \sum_{k=1}^{\infty} \frac{\mu_{k-1}}{x^k} - \Theta(x), \end{aligned}$$

also

$$\Theta(x) = \underbrace{\sum_{k=1}^{\infty} \frac{\mu_{k-1}}{x^k}}_{=:M(x)} - \sum_{j=0}^n \frac{w_j^*}{x-x_j}. \tag{A.35}$$

Da die Quadraturformel interpolatorisch ist, ist auf alle Fälle $\theta_j = 0, j = 0, \dots, n$, das heißt,

$$\Theta(x) = \sum_{j=n+1}^{\infty} \frac{\theta_j}{x^{j+1}} = x^{-(n+2)} (\theta_{n+1} + \theta_{n+2}x^{-1} + \dots)$$

und damit besitzt die Funktion $x \mapsto \omega(x) \Theta(x)$ nur *negative* Exponenten. Nun ist, mit (A.35),

$$\begin{aligned} \omega(x) \Theta(x) &= \omega(x) \left(M(x) - \sum_{j=0}^n \frac{w_j^*}{x-x_j} \right) = \omega(x) M(x) - \omega(x) \sum_{j=0}^n \frac{w_j^*}{x-x_j} \\ &= \underbrace{\omega_1(x)}_{\in \Pi_n} + \omega_2(x) - \underbrace{\sum_{j=0}^n w_j^* \frac{\omega(x)}{x-x_j}}_{\in \Pi_n}. \end{aligned}$$

Also ergibt der polynomiale Teil die Identität

$$\omega_1(x) = \omega(x) \sum_{j=0}^n \frac{w_j^*}{x-x_j} \tag{A.36}$$

und, weil $\omega(x) \Theta(x)$ nur negative Potenzen von x enthält, erhalten wir

$$\omega_2(x) = \omega(x) \Theta(x). \tag{A.37}$$

Kettenbrüche. Jetzt verwenden wir die Bernoullische Formel aus Aufgabe A.3, um die Koeffizienten v_j und w_j der Kettenbruchentwicklung

$$M(x) = \sum_{j=1}^{\infty} \mu_{j-1} x^{-j} = \frac{v_1}{w_1} + \frac{v_2}{w_2} + \dots$$

zu bestimmen, und zwar ist, mit

$$c_j = \sum_{k=1}^j \mu_{k-1} x^{-k}, j \in \mathbb{N},$$

zuerst einmal $v_1 = \mu_0$ und $w_1 = x$ sowie

$$v_j = \frac{c_{j-1} - c_j}{c_{j-1} - c_{j-2}} = \frac{-\mu_{j-1} x^{-j}}{\mu_{j-2} x^{-j+1}} = -\frac{\mu_{j-1}}{\mu_{j-2}} x^{-1}, \quad j > 1,$$

sowie

$$w_j = \frac{c_j - c_{j-2}}{c_{j-1} - c_{j-2}} = \frac{\mu_{j-1} x^{-j} + \mu_{j-2} x^{-j+1}}{\mu_{j-2} x^{-j+1}} = \frac{\mu_{j-1}}{\mu_{j-2}} x^{-1} + 1, \quad j > 1.$$

Multiplizieren wir nun²¹⁴ jeweils v_j und w_j mit x , dann erhalten wir daß $v_1 = 1, w_1 = x$ und

$$v_j = -\frac{\mu_{j-1}}{\mu_{j-2}} \quad \text{und} \quad w_j = x + \frac{\mu_{j-1}}{\mu_{j-2}} \quad j \geq 2. \quad (\text{A.38})$$

Insbesondere ist $V_j \in \Pi_{j-1}$ und $W_j \in \Pi_j, j \in \mathbb{N}$, und für alle $m \in \mathbb{N}$ ist nach Teil 2 von Aufgabe A.3

$$M(x) - \frac{V_m(x)}{W_m(x)} = \sum_{k=m}^{\infty} \left(\frac{V_{k+1}(x)}{W_{k+1}(x)} - \frac{V_k(x)}{W_k(x)} \right) = \sum_{k=m}^{\infty} (-1)^k \frac{v_1 \cdots v_{k+1}}{W_k(x) W_{k+1}(x)}.$$

Da man für $p \in \Pi_j$ die *Laurententwicklung*

$$\frac{1}{p} = \sum_{k=j}^{\infty} \alpha_k x^{-k}$$

erhält²¹⁵, ist also

$$M(x) - \frac{V_{n+1}(x)}{W_{n+1}(x)} = \sum_{k=n+1}^{\infty} (-1)^k \underbrace{\frac{v_1 \cdots v_{k+1}}{W_k(x) W_{k+1}(x)}}_{\in \Pi_{2k+1}} = \sum_{k=2n+2}^{\infty} \frac{\alpha_k}{x^{k+1}},$$

²¹⁴Das ändert den Wert des Kettebruchs nicht.

²¹⁵Wie man das beweist? Einfach mit p durchmultiplizieren, das liefert dann die Bedingungsgleichungen für die $\alpha_j, j \geq n$.

oder

$$M(x) W_{n+1}(x) = V_{n+1}(x) + W_{n+1}(x) \sum_{k=2n+2}^{\infty} \frac{\alpha_k}{x^{k+1}}. \quad (\text{A.39})$$

Exaktheit der Quadraturformel. Da W_{n+1} durch eine Drei-Term-Rekursionsformel definiert ist, hat es $n + 1$ verschiedene reelle Nullstellen, x_j^* , $j = 0, \dots, n$. Dann ist

$$\omega^*(x) := (x - x_0^*) \cdots (x - x_n^*)$$

ein Vielfaches von W_{n+1} und es ist

$$\omega_1^*(x) + \omega_2^*(x) = M(x) \omega^*(x) = V_{n+1}(x) + \omega^*(x) \sum_{k=2n+2}^{\infty} \frac{\alpha_k}{x^{k+1}},$$

das heißt, wenn wir nur die Terme mit negativem Exponenten betrachten,

$$\omega_2^*(x) = \omega^*(x) \sum_{k=2n+2}^{\infty} \frac{\alpha_k}{x^{k+1}}$$

und damit liefert (A.37), daß für diese Knotenwahl

$$\sum_{k=1}^{\infty} \frac{\theta_{k-1}}{x^k} = \Theta(x) = \frac{\omega_2^*(x)}{\omega^*(x)} = \sum_{k=2n+1}^{\infty} \frac{\alpha_{k-1}}{x^k},$$

also, nach Koeffizientenvergleich, daß $\theta_0 = \cdots = \theta_{2n+1} = 0$. Das war's!

*Uns ist in alten mæren
wunders viel geseit
von Helden lobebæren
von grôzer arebeit*

Das Nibelungenlied

Literatur | A

- [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 probabilités*, Commun. Soc. Math. Kharkov **13** (1912), 1–2.
- [4] C. de Boor, *A practical guide to splines*, Springer-Verlag, New York, 1978.
- [5] C. de Boor, *Splinefunktionen*, Lectures in Mathematics, ETH Zürich, Birkhäuser, 1990.
- [6] C. G. Broyden, *A class of methods for solving nonlinear simultaneous equations*, Math. Comp. **19** (1965), 577–593.
- [7] C. K. Chui, *Multivariate splines*, CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 54, SIAM, 1988.
- [8] P. J. Davis, *Interpolation and approximation*, Dover Books on Advanced Mathematics, Dover Publications, 1975.
- [9] C. F. Gauss, *Methodus nova integralium valores per approximationem inveniendi*, Commentationes societate regiae scientiarum Gottingensis recentiores **III** (1816).
- [10] W. Gautschi, *Numerical analysis. an introduction*, Birkhäuser, 1997.
- [11] W. Givens, *Computation of plane unitary rotations transforming a matrix to triangular form*, SIAM J. Appl. Math. **6** (1958), 26–50.

- [12] D. Goldberg, *What every computer scientist should know about floating-point arithmetic*, ACM Computing Surveys **23** (1991), 5–48.
- [13] G. Golub and C. F. van Loan, *Matrix computations*, The Johns Hopkins University Press, 1983.
- [14] G. H. Golub, *Numerical methods for solving linear least squares problems*, Numer. Math. **7** (1965), 206–216.
- [15] G. H. Golub (ed.), *Studies in numerical analysis*, MAA Studies in Mathematics, vol. 24, The Mathematical Association of America, 1984.
- [16] H. Heuser, *Lehrbuch der Analysis. Teil 2*, 2. ed., B. G. Teubner, 1983.
- [17] N. J. Higham, *Accuracy and stability of numerical algorithms*, SIAM, 1996.
- [18] A. S. Householder, *Unitary triangularization of a nonsymmetric matrix*, J. Assoc. Comp. Mach. **5** (1958), 339–342.
- [19] E. Isaacson and H. B. Keller, *Analysis of Numerical Methods*, John Wiley & Sons, 1966.
- [20] C. G. J. Jacobi, *Über ein leichtes Verfahren, die in der Theorie der Säkularstörungen vorkommenden Gleichungen numerisch aufzulösen*, Crelle's Journal **30** (1846), 51–94.
- [21] D. Jörgensen, *Der Rechenmeister*, Rütten & Loenig, 1999, Roman.
- [22] S. Karlin, *Total positivity*, Stanford University Press, Stanford CA, 1968.
- [23] E. Kreyszig, *Introductory functional analysis with applications*, John Wiley & Sons, 1978.
- [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] W. Popp, *Wege des exakten denkens. vier jahrtausende mathematik*, Weltbild Verlag, 1987, Originalausgabe Franz Ehrenwirth Verlag, 1981.
- [32] 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.
- [33] T. Sauer, *Computeralgebra*, Vorlesungsskript, Justus–Liebig–Universität Gießen, 2001.
- [34] ———, *Optimierung*, Vorlesungsskript, Universität Passau, 2013.
- [35] I. J. Schoenberg, *Cardinal spline interpolation*, CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 12, SIAM, 1973.
- [36] L. L. Schumaker, *Spline funtions: Basic theory*, Pure and Applied Mathematics: A Wiley–Interscience Series of Texts, Monographs and Tracts, John Wiley & Sons, 1981.
- [37] H. R. Schwarz, *Numerische Mathematik*, B. G. Teubner, Stuttgart, 1988.
- [38] L.E. Sigler, *Fibonacci's Liber Abaci. Leoanardo Pisano's book of calculation*, Springer, 2002.
- [39] P. Spellucci, *Numerische Verfahren der nichtlinearen Optimierung*, Internationale Schriftenreihe zu Numerischen Mathematik, Birkhäuser, 1993.
- [40] J. Stoer, *Einführung in die Numerische Mathematik I*, 4 ed., Heidelberger Taschenbücher, Springer Verlag, 1983.
- [41] J. Stoer and R. Bulirsch, *Einführung in die Numerische Mathematik II*, 2 ed., Heidelberger Taschenbücher, Springer Verlag, 1978.
- [42] K. Weierstraß, *Über die analytische Darstellbarkeit sogenannter willkührlicher Funktionen reller Argumente*, Sitzungsber. Kgl. Preuss. Akad. Wiss. Berlin (1885), 633–639, 789–805.
- [43] J. H. Wilkinson, *The evaluation of zeros of ill–conditioned polynomials I*, Numer. Math. **1** (1959), 150–166.
- [44] ———, *The evaluation of zeros of ill–conditioned polynomials II*, Numer. Math. **1** (1959), 167–180.
- [45] ———, *Error analysis of floating point computation*, Numer. Math. **2** (1960), 319–340.

- [46] ———, *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.
- [47] Yuan Xu, *Common zeros of polynomials in several variables and higher dimensional quadrature*, Pittman Research Monographs, Longman Scientific and Technical, 1994.

- 1-Norm, 31
- ∞ -Norm, 32
- p-Normen, 32

- a-priori-Abschätzungen, 27
- abgebrochene Potenz, 171
- absoluter Fehler, 12
- adaptiv, 201
- Addition, 16
- Additionstheoreme, 142
- affine Kombination, 124
- Aitken–Neville–Algorithmus, 130
- Akkumulator, 16
- Algebra, 122
- Algorithmus von de Casteljau, 125
- Auslöschung, 111

- B-Spline, 154
- Bézierkurve, 125
- Banachscher Fixpunktsatz, 82
- bandiert, 74, 154
- baryzentrische Koordinaten, 124
- Basis, 13
- Bernstein–Bézier–Basispolynome, 125
- Bernsteinpolynome, 125, 216
- Betragsmatrix, 56
- Biegeenergie, 159
- binär, 13
- Bisektionsverfahren, 176
- Blossoming Principle, 165
- Broyden–Verfahren, 190

- Cauchyfolge, 82, 186
- CAVALIERI, 198
- Cholesky–Zerlegung, 73, 74
- compensated summation, 27
- Cramersche Regel, 6

- Determinante, 41
- diagonal, 101
- Diagonaldominanz, 92
- Distanz, 14
- dividierte Differenz, 134
- Doolittle–Methode, 42
- Doolittle–Verfahren, 74
- doppelte Nullstellen, 190
- Drehung, 118
- Drei–Term–Rekursionsformel, 208
- Dreiecksungleichung, 31
- dualer Vektor, 34
- dunnbesetzte Matrizen, 88

- Effizienz, 4
- Einschlussverfahren, 176
- Einzelschrittverfahren, 87
- Elchataym, 179
- Energienorm, 159
- Erwartungswert, 7
- Euklidischer Abstand, 31
- Exaktheitsgrad, 193, 196
- Exponentenbereich, 13

- Fehlerfunktional, 193
- Fibonacci–Zahlen, 181
- Fixpunkt, 81, 82, 186
- Frobenius–Norm, 116
- Frobeniusnorm, 32
- Funktional, 192

- Gaus–Elimination, 42, 61, 80
- Gaus–Formel, 196, 203
- Gaus–Matrix, 48
- Gaus–Quadratur, 193
- Gaus–Seidel–Iteration, 87
- Gaus–Transformation, 48

- Gesamtschrittverfahren, 87
 Gewichte, 193
 Gewichtsfunktion, 192
 Givens–Rotationen, 117
 gleichmasig stetig, 217
 Gleitkommaoperationen, 16
 Gleitkommazahl, 14
 Gleitkommazahlen, 13
 global konvergent, 176
 Goldener Schnitt, 181
 Grad, 121
 Gram–Schmidt, 204
 Guard Digit, 18
 Gutefunktional, 96
- Halbordnung, 56
 Hankelmatrix, 205
 Hauptminore, 41
 Hauptuntermatrix, 41
 Hermite–Interpolant, 206
 hermitesche Matrix, 73
 Heron–Verfahren, 174
 Hesse–Matrix, 189
 Householder–Matrix, 105
 Householder–Transformation, 112
 Householder–Vektor, 106
- innere Knoten, 149
 instabil, 7
 Integral, 192
 Interpolation, 30, 196
 Interpolationsoperator, 133
 Interpolationsproblem, 30, 129
 interpolatorisch, 196
 isolierte Nullstelle, 219
 Iterationsmatrix, 82
 iterative refinement, 68
 iteratives Verfahren, 81
 Iteratives Verfeinern, 68
- Jacobi–Matrix, 118, 186
 Jacobi–Verfahren, 87
 Jordan–Normalform, 84
 kanonische Koordinatenprojektion, 115
- Keplersche Fasregel, 198
 Knoten, 149, 193, 197
 Knotenfolge, 149
 Knotenpolynom, 197
 Kollokationsmatrix, 154, 155, 172
 komponentenweiser Rückwärtsfehler, 25
 Konditionszahl, 25, 34, 104
 konsistent, 33
 Kontraktion, 82, 186
 Kontrollpolygon, 125
 Kontrollpunkte, 125
 Konvergenzordnung, 86, 176
 Konvexkombination, 124, 128, 203, 219
 kubischer Spline, 159
- Lagrange–Darstellung, 130
 Lagrange–Multiplikatoren, 96
 Least–Squares–Lösung, 98, 104, 115
 Leibniz–Formel, 171
 Leibniz–Regel, 6
 Leitkoeffizient, 141
 Leitkoeffizienten, 138
 Leitterm, 141, 205
 lineare Konvergenz, 86
 Lineares Gleichungssystem, 30
 lokal konvergent, 176
- Manhattan–Norm, 31
 Mantisse, 13
 Mantissenlänge, 13
 Matlab, 36, 152
 Matrix, 30
 Matrix–Norm, 32
 Median, 98
 Milne–Regel, 198
 Momente, 193
 Momentenmatrix, 205
 monische orthogonale Polynome, 205
 Monom, 205
 Monome, 121
 monotone Matrixnorm, 58
 Moore–Penrose–Bedingungen, 103
 Moore–Penrose–Inverse, 103
 Multiplikation, 16

- Nachiteration, 68
 Näherung, 4
 natürlicher Spline, 159
 natürlicher Splineinterpolant, 157
 Newton–Cotes–Formeln, 197
 Newton–Darstellung, 133
 nichtlineares Gleichungssystem, 188
 nichtnegative Teilung der Eins, 164
 nilpotent, 92
 Norm, 31
 Normalgleichungen, 98
 normalisiert, 14
 normiert, 106
 Nullspalte, 115

 obere Dreiecksmatrix, 38, 49
 Octave, 36, 152
 Operatornorm, 33, 83, 187
 Optimierungsproblem, 96
 Optische Tomographie, 95
 orthogonal, 101, 204
 orthogonal invariant, 105
 orthogonale Matrix, 113
 orthogonales Polynom, 204
 Orthogonalisierungsverfahren, 204
 Oslo–Algorithmus, 170

 parallelisierbar, 88
 Pascal–Matrix, 78
 Permutationsmatrix, 59
 Pivotelement, 59
 Pivotsuche, 115
 Polynom, 121
 polynomiale Kurve, 122
 positiv definit, 72
 positiv semidefinit, 72
 Positive Homogenität, 31
 Positivität, 31
 Projektion, 60, 196
 Pseudoinverse, 103
 Pulcherrima, 198
 Pyramidenschema, 130

 Quadraturformel, 193

 Quadratwurzel, 174

 Randknoten, 149
 Rechtsdreiecksmatrix, 113
 Regula Falsi, 179
 Rekursionsformel, 208
 relativer Fehler, 12
 Residuenvektor, 99
 Rotation, 118
 Rucksubstitution, 39, 55, 115, 211
 Rückwärtsfehler, 24, 56
 Rundungsfehlereinheit, 14
 runnig error analysis, 27

 Satz von Hahn–Banach, 34
 Satz von Weierstrass, 122
 Schoenberg–Whitney, 154
 Schoenberg–Whitney–Bedingung, 172
 Sekantenverfahren, 181
 Signatur, 217
 SIMPSON, TH., 198
 Simpson–Regel, 198
 Singular Value Decomposition, 101
 Singulare–Werte–Zerlegung, 101
 Singularwerte, 101
 Skalarprodukt, 22, 204
 Skalierung, 67
 Spaltenpivotsuche, 61, 64
 Spaltensummennorm:, 33
 Spektralradius, 83, 100, 186
 Spektrum, 83
 Spiegelung, 106
 Spline, 159
 Splinekurve, 152
 Splineraum, 165
 Stabilität, 116
 Standardabweichung, 7
 Standardmodell der Gleitkommaarithmetik,
 20
 Störungsanfälligkeit, 4
 Straklatte, 159
 strikte Quadratpositivität, 192
 Summationsalgorithmus, 27
 SVD, 101

symmetrisch, 72

Tangente, 184

total nichtnegativ, 78, 154

total positiv, 78

Totalpivotsuche, 61, 64

Trapezregel, 198

Tschebyscheff–Polynom, 141

Tschebyscheffknoten, 194

überbestimmt, 30

Überlauf, 13

unit roundoff, 14

unitar, 101

unterbestimmt, 30

untere Dreiecksmatrix, 38

Unterlauf, 13

Varianz, 7

Vektor, 30

Vektornorm, 187

Vektornormen, 32

Vektorraum, 32

Verfeinerung, 170

Vertauschungsmatrix, 59

vertraglich, 33

Vielfachheit, 149

Vorwärtselimination, 39, 55, 211

Vorzeichenwechsel, 217

Weddle–Regel, 198

Worst–Case–Fehler, 32

Zeilenform, 49

Zeilenpivotsuche, 61, 64

Zeilensummennorm:, 33

zeilenweise diagonaldominant, 92

Zerlegung, 55

zusammengesetzten Simpson–Regel, 202

zweizeiliges Horner Schema, 230