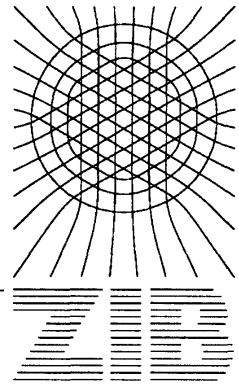
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Heilbronner Str. 10, D-1000 Berlin 31



P. Deuflhard A. Hohmann

Einführung in die Numerische Mathematik

Skriptum zur Vorlesung, FU Berlin, SS 90

Technical Report TR 90-6 (September 1990)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Heilbronner Str. 10
1000 Berlin 31
Verantwortlich: Dr. Klaus André
Umschlagsatz und Druck: Rabe KG Buch-und Offsetdruck Berlin

ISSN 0933-789X

Vorwort

Das vorliegende Skriptum entstand aus einer Vorlesung, die ich im SS 90 an der Freien Universität Berlin im Fachbereich Mathematik gehalten habe. Sie wurde von Herrn Dipl.–Math. A. Hohmann ausgehend von meinen Handnotizen wesentlich überarbeitet, umorganisiert und substantiell ergänzt. Darüberhinaus haben Herr Dipl.–Math. M. Klatte (Kapitel 5.3) sowie Herr Dipl.–Math. M. Wulkow (Kapitel 6.5) wichtige Beiträge geleistet.

Die Vorlesung war gedacht als Einführung in die Numerische Mathematik für Studierende der Fachrichtungen Mathematik, Informatik und Naturwissenschaften. Grundidee war die transparente Darstellung wichtiger moderner Konzepte am jeweils einfachsten Problemtyp, um möglichst geringe anderweitige Vorkenntnisse voraussetzen zu müssen. Trotz dieser Beschränkung enthält das Skriptum zum Teil unveröffentlichten Stoff und berücksichtigt Originalliteratur jüngeren Datums, die bisher noch keinen Eingang in Lehrbücher gefunden hat. Auch bei relativ bekanntem Lehrbuchstoff unterscheidet sich der vorliegende Text in der Sichtweise. Der Blick ist im wesentlichen auf die Entwicklung effektiver Algorithmen gerichtet — was natürlich Theorie einschließt, die für die algorithmische Realisierung relevant ist. Die hier vorgestellte Numerische Mathematik ist — innerhalb des relativ jungen Gebietes *Scientific Computing* — vernetzt mit Informatik und Naturwissenschaften.

Mein herzlicher Dank geht an Frau E. Bräuer, die in bewundernswerter Sorgfalt und Schnelle das Skriptum in TEX erstellt hat. Herrn Dipl.–Math. F. A. Bornemann danke ich ebenfalls herzlich für zahlreiche Diskussionen.

Peter Deuflhard

Inhalt

Symbole

$\mathbf{R}$	reelle Zahlen
$\mathbf{C}$	komplexe Zahlen
$\mathbf{N}$	natürliche Zahlen, $\mathbf{N} = \{0, 1, 2, \ldots\}$
O, o	Landau-Symbole
$\doteq$	in führender Ordnung gleich
$\mathrm{Mat}_{m,n}(\mathbf{K})$	$m \times n$-Matrizen mit Koeffizienten in $\mathbf{K}$
$\mathrm{Mat}_n(\mathbf{K})$	$n \times n$-Matrizen mit Koeffizienten in $\mathbf{K}$
$B_\rho(x)$	offene Kugel $\{y \mid \|x - y\| < \rho\}$ mit Radius ρ um x
$\bar{B}_\rho(x)$	abgeschlossene Kugel $\{y \mid \|x - y\| < \rho\}$ mit Radius ρ um x
$\mathrm{GL}(n)$	invertierbare $n \times n$-Matrizen $\{A \in \mathrm{Mat}_n(\mathbf{R}) \mid \det(A) \neq 0\}$
$\mathrm{O}(n)$	orthogonale $n \times n$-Matrizen $\{A \in \mathrm{GL}(n) \mid AA^T = A^T A = I\}$
$\mathbf{P}_n$	Polynome vom Grad kleiner oder gleich n mit Koeffizienten in $\mathbf{R}$
$\mathbf{P}_d^n$	verallgemeinerte Polynome vom Grad kleiner oder gleich n mit Koeffizienten in $\mathbf{R}^d$

1 Gauß'sche Dreieckszerlegung

Wir beginnen mit einem Verfahren zur Auflösung eines linearen Gleichungssystems. Die Gauß'sche Dreieckszerlegung basiert auf Überlegungen, die Carl Friedrich Gauß (1777 - 1855) 1801 in seinen "Disquisitiones arithmeticae" veröffentlicht hat. Er berechnete mit den dort beschriebenen Methoden die Bahn des Planetoiden Ceres, dessen Wiederauffindung im gleichen Jahr an der von ihm vorausberechneten Stelle gelang (was ihm einigen Ruhm einbrachte).

1.1 Einführung

Zu lösen ist ein System von n linearen Gleichungen

$$Ax = b, \quad A\ (n,n)\text{-Matrix}, \quad b, x\ n\text{-Vektoren}$$

Wir schreiben dabei bewußt "A (n,n)-Matrix" und "x n-Vektor" anstelle der "mathematischeren" Notation "$A \in \mathrm{Mat}_n(\mathbf{R})$" und "$x \in \mathbf{R}^n$", um von vornherein klarzustellen, daß die reellen Zahlen eine rein theoretische Konstruktion sind, die auf keinem Rechner (und auf keinem Blatt Papier) realisiert werden können. Jede Zahl kann "real" nur durch eine endliche Ziffernfolge repräsentiert werden. Mathematisch gesprochen bilden diese Zahlen noch nicht einmal einen Körper, geschweige denn einen vollständigen. Für eine Existenz- und Eindeutigkeitsaussage greifen wir aber auf die reellen Zahlen und das folgende Resultat aus der linearen Algebra zurück.

Satz 1.1.1 *Sei* $A \in \mathrm{Mat}_n(\mathbf{R})$ *eine relle quadratische Matrix mit* $\det(A) \neq 0$ *und* $b \in \mathbf{R}^n$. *Dann existiert genau ein* $x \in \mathbf{R}^n$, *so daß* $Ax = b$.

Falls $\det(A) \neq 0$, läßt sich die Lösung $x = A^{-1}b$ mit der Cramer'schen Regel berechnen. Wir sehen hier bereits eine wichtige Eigenschaft eines "guten" Algorithmus, nämlich die Verbindung der Existenz- und Eindeutigkeitsaussage mit dem Rechenverfahren, welches nur im Falle $\det(A) \neq 0$ eine Lösung ergibt (Division durch $\det(A)$).

Von einem guten Rechenverfahren erwarten wir aber auch, daß es die gestellte Aufgabe möglichst *effizient*, d.h. mit möglichst geringem Aufwand (an Rechenoperationen) löst. Den (theoretisch) minimalen Rechenaufwand zur Lösung eines Problems bezeichnen wir als *Komplexität* des Problems. Ein Algorithmus ist umso effektiver je näher der benötigte Rechenaufwand an der Komplexität des Problems liegt.

Aufwand zur Berechnung von $\det(A)$:

a) mit der Cramer'schen Regel: $\sim n!$ Operationen

b) mit der Laplace'schen Determinantenformel: $\sim 2^n$ Operationen

Wie wir sehen werden, sind alle folgenden Verfahren bereits für $n \geq 3$ effektiver als die Cramer'sche Regel, so daß diese nur für $n = 2$ in Frage kommt.

Die Schreibweise $x = A^{-1}b$ könnte den Gedanken aufkommen lassen, zur Berechnung der Lösung von $Ax = b$ zunächst die inverse Matrix A^{-1} zu berechnen und diese dann

auf b anzuwenden. Die Berechnung von A^{-1} beinhaltet jedoch die Schwierigkeiten der Lösung von $Ax = b$ für sämtliche b. Wir werden im zweiten Kapitel sehen, daß die Berechnung von A^{-1} "bösartig" sein kann, auch wenn sie für spezielle b "gutartig" ist. Bei der Notation $x = A^{-1}b$ handelt es sich daher nur um eine formale Schreibweise, die nichts mit der tatsächlichen Berechnung der Lösung x zu tun hat.

1.2 Auflösung gestaffelter Systeme

Sei R eine obere Dreiecksmatrix. Wir wollen das System

$$Rx = z \tag{1.2.1}$$

bzw. elementweise

$$\begin{array}{lllllll} r_{11}x_1 & + & r_{12}x_2 & + \ \ldots & + & r_{1n}x_n & = & z_1 \\ & & r_{22}x_2 & + \ \ldots & + & r_{2n}x_n & = & z_2 \\ & & & \ddots & & \vdots & & \vdots \\ & & & & & r_{nn}x_n & = & z_n \end{array}$$

lösen und erhalten x durch rekursive Auflösung, beginnend mit der Zeile n:

$$\begin{array}{lll} x_n & := & z_n/r_{nn} \qquad\qquad\qquad , \ \text{falls} \ r_{nn} \neq 0 \\ x_{n-1} & := & (z_{n-1} - r_{n-1,n}x_n)/r_{n-1,n-1} \quad , \ \text{falls} \ r_{n-1,n-1} \neq 0 \\ \vdots & & \\ x_1 & := & (z_1 - r_{12}x_2 - \ldots - r_{1n}x_n)/r_{11} \quad , \ \text{falls} \ r_{11} \neq 0 \end{array}$$

Nun gilt für die obere Dreiecksmatrix R, daß

$$\begin{array}{lll} \text{a)} & \det R & = \ r_{11} \cdot \ldots \cdot r_{nn} \\ \text{b)} & \det R & \neq \ 0 \longleftrightarrow \forall i = 1, \ldots, n \ \ r_{ii} \neq 0 \end{array}$$

Der angegebene Algorithmus ist also genau dann anwendbar, wenn $\det R \neq 0$, also unter der Bedingung des Existenz- und Eindeutigkeitssatzes. Für den Rechenaufwand ergibt sich:

a) pro Zeile i in (1.1.3) :

   $(i-1)$ Multiplikationen, $(i-1)$ Additionen, 1 Division

b) insgesamt für Zeile n bis 1 :

   $\sum_{i=1}^{n}(i-1) = n(n-1)/2 \doteq n^2/2$ Multiplikationen und Additionen

Dabei steht das Zeichen "$\doteq$" für "gleich bis auf Terme niedrigerer Ordnung".

Vollkommen analog läßt sich ein gestaffeltes Gleichungssystem der Form

$$Lx = z \tag{1.2.2}$$

mit einer unteren Dreiecksmatrix L lösen, indem man jetzt in der ersten Zeile beginnt und sich bis zur letzten Zeile durcharbeitet. Diese Auflösung gestaffelter Systeme heißt im Fall von (1.2.1) *Rückwärtssubstitution* und im Fall von (1.2.2) *Vorwärtssubstitution.* Der Name Substitution, d. h. Ersetzung, rührt daher, daß der Vektor der rechten Seite jeweils komponentenweise ersetzt werden kann.

$$\begin{aligned}
&(z_1, z_2, \ldots, z_{n-1}, z_n)\\
&(z_1, z_2, \ldots, z_{n-1}, x_n)\\
&\qquad\vdots\\
&(z_1, x_2, \ldots, x_{n-1}, x_n)\\
&(x_1, x_2, \ldots, x_{n-1}, x_n)
\end{aligned}$$

1.3 Die Gauß'sche Eliminationsmethode

Wir gehen nun wieder zurück zu dem linearen Gleichungssystem $Ax = b$ für allgemeine (n, n)-Matrizen A, d.h.

$$\begin{array}{lcllcl}
a_{11}x_1 & + & a_{12}x_2 & +\ldots+ & a_{1n}x_n & = & b_1\\
a_{21}x_1 & + & a_{22}x_2 & +\ldots+ & a_{2n}x_n & = & b_2\\
\vdots & & \vdots & & \vdots & & \vdots\\
a_{n1}x_1 & + & a_{n2}x_2 & +\ldots+ & a_{nn}x_n & = & b_n
\end{array} \tag{1.3.1}$$

und versuchen, dieses Gleichungssystem in ein gestaffeltes umzuformen.

Die erste Zeile muß dazu nicht verändert werden. Die restlichen Zeilen sind so zu behandeln, daß die Koeffizienten vor x_1 verschwinden und ein System der Art

$$\begin{array}{lcllcl}
a_{11}x_1 & + & a_{12}x_2 & +\ldots+ & a_{1n}x_n & = & b_1\\
 & & a'_{22}x_2 & +\ldots+ & a'_{2n}x_n & = & b'_2\\
 & & \vdots & & & & \vdots\\
 & & a'_{n2}x_2 & +\ldots+ & a'_{nn}x_n & = & b'_n
\end{array} \tag{1.3.2}$$

entsteht. Haben wir das erreicht, so können wir dasselbe Verfahren auf die letzten $n-1$ Zeilen anwenden und so rekursiv das gewünschte gestaffelte System erhalten. Daher genügt es, den ersten Eliminationsschritt von (1.3.1) zu (1.3.2) zu untersuchen. Wir setzen voraus, daß $a_{11} \neq 0$. Um den Term $a_{i1}x_1$ in Zeile i $(i = 2, \ldots, n)$ zu eliminieren, substrahieren wir von der Zeile i ein Vielfaches der unveränderten Zeile 1, die wir auch *Pivotzeile* nennen, d.h.

$$\text{Zeile } i \text{ neu} := \text{Zeile } i - \ell_{i1} \cdot \text{Zeile } 1$$

oder explizit

$$\underbrace{(a_{i1} - \ell_{i1}a_{11})}_{=0} x_1 + \underbrace{(a_{i2} - \ell_{i1}a_{12})}_{=a'_{i2}} x_2 + \cdots + \underbrace{(a_{in} - \ell_{i1}a_{1n})}_{=a'_{in}} x_n = \underbrace{b_i - \ell_{i1}b_1}_{=b'_i}$$

Aus $a_{i1} - \ell_{i1}a_{11} = 0$ folgt sofort, daß wir

$$\ell_{i1} := a_{i1}/a_{11}$$

wählen müssen und die Eliminationsmethode nur ausführbar ist, falls die sukzessive auftretenden Diagonalelemente $a_{11}, a'_{22}, a''_{33}$, die sogenannten *Pivotelemente*, nicht verschwinden.

Wir erhalten so eine Kette von Matrizen

$$A = A^{(1)} \to A^{(2)} \to \ldots \to A^{(n)} =: R$$

und äquivalenten Gleichungssystemen

$$A^{(k)}x = b^{(k)}, \quad k = 2, \ldots, n$$

wobei R eine obere Dreiecksmatrix ist und für $k = 1, \ldots, n-1$ gilt:

$$\begin{aligned} a_{ij}^{(k+1)} &:= a_{ij}^{(k)} - \ell_{ik}a_{kj}^{(k)} \quad , \quad i,j = k+1, \ldots, n \\ b_i^{(k+1)} &:= b_i^{(k)} - \ell_{ik}b_k^{(k)} \quad , \quad i = k+1, \ldots, n \\ \ell_{ik} &:= a_{ik}^{(k)}/a_{kk}^{(k)} \end{aligned}$$

In Matrixschreibweise lassen sich diese Operationen durch die Multiplikation von links mit einer unipotenten (alle Diagonalelemente 1) unteren Dreiecksmatrix beschreiben, d.h.

$$A^{(k+1)} = L_k A^{(k)}, \quad b^{(k+1)} = L_k b^{(k)}$$

wobei L_k die Matrix

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

In dieser Schreibweise erhalten wir das zu $Ax = b$ äquivalente gestaffelte Gleichungssystem $Rx = z$ durch

$$\begin{aligned} &\text{a)} \quad R := A^{(n)} = L_{n-1} \cdot \ldots \cdot L_1 A \\ &\text{b)} \quad z := b^{(n)} = L_{n-1} \cdot \ldots \cdot L_1 b \end{aligned}$$

Setzen wir

$$L := L_1^{-1} \cdot \ldots \cdot L_{n-1}^{-1} = \begin{bmatrix} 1 & & & & \\ \ell_{21} & 1 & & & \\ \ell_{31} & \ell_{32} & 1 & & \\ \vdots & & \ddots & \ddots & \\ \ell_{n1} & & \ldots & \ell_{n,n-1} & 1 \end{bmatrix}$$

(wobei man beachte, daß die unipotenten unteren Dreiecksmatrizen eine Untergruppe von $SL(n)$ bilden, siehe Aufgabe 1.6), so folgt

$$A = LR, \quad b = Lz \; .$$

Die Darstellung von A als Produkt $A = LR$ einer unipotenten unteren Dreiecksmatrix L und einer oberen Dreiecksmatrix heißt *LR-Zerlegung* von A. Falls sie existiert, so sind L und R eindeutig bestimmt (siehe Aufgabe 1.2).

Zusammenfassung der Gauß-Elimination

a)	$A = LR$	Dreieckszerlegung (= Elimination)
		R obere, L unipotente untere Dreiecksmatrix
b)	$Lz = b$	Vorwärtssubstitution
c)	$Rx = z$	Rückwärtssubstitution

Der Speicheraufwand für die Gauß-Elimination beträgt $n \cdot (n+1)$ Speicherplätze, d.h. genauso viele wie für die Eingabe benötigt werden, wenn man die Operationen innerhalb der Matrix ausführt.

Für den Rechenaufwand ergibt sich (wir zählen nur die Multiplikationen):

1. $\sim \sum_{k=1}^{n-1} k^2 \doteq n^3/3$ Multiplikationen für a)
2. jeweils $\sim n^2/2$ Multiplikationen für b) und c)

Der Hauptaufwand besteht also in der LR-Zerlegung , die für verschiedene rechte Seiten $b_1, \ldots, b_k$ aber nur einmal durchgeführt werden muß.

1.4 Pivot-Strategien

Wie man bereits an dem einfachen Beispiel

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \det A = -1 \; , \quad a_{11} = 0$$

sieht, gibt es Fälle, bei denen die Dreieckszerlegung versagt, obwohl $\det A \neq 0$. Eine Permutation der Zeilen führt jedoch zur denkbar einfachsten LR-Zerlegung

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \longrightarrow \quad \bar{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I = LR \text{ mit } L = R = I \; .$$

Bei der rechnerischen Realisierung des Gauß'schen Eliminationsverfahrens können bereits Schwierigkeiten bei "zu kleinen" Pivotelementen entstehen.

Beispiel 1.4.1 Wir berechnen die Lösung des Gleichungssystems

$$\begin{aligned} &(a) \quad 1.00 \cdot 10^{-4} x_1 + 1.00 x_2 = 1.00 \\ &(b) \quad 1.00 x_1 + 1.00 x_2 = 2.00 \end{aligned}$$

auf einem Rechner, der der Einfachheit halber mit der Basis 10 und der Mantissenlänge 3 arbeite. Ergänzen wir die Zahlen durch Nullen, so erhalten wir als "exakte" Lösung auf vier gültige Ziffern

$$x_1 = 1.001 \quad x_2 = 1.000$$

auf drei gültige also

$$x_1 = 1.00 \quad x_2 = 1.00$$

Wir führen nun die Gauß-Elimination mit unserem Rechner, d.h. auf 3 gültige Ziffern, aus:

$$\ell_{21} = \frac{a_{21}}{a_{11}} = \frac{1.00}{1.00 \cdot 10^{-4}} = 1.00 \cdot 10^4$$

$$(1.00 - 1.00 \cdot 10^4 \cdot 1.00 \cdot 10^{-4}) x_1 + (1.00 - 1.00 \cdot 10^4 \cdot 1.00) x_2$$
$$= 2.00 - 1.00 \cdot 10^4 \cdot 1.00$$

Wir erhalten das gestaffelte System

$$1.00 \cdot 10^{-4} x_1 + 1.00 x_2 = 1.00$$

$$-1.00 \cdot 10^4 x_2 = -1.00 \cdot 10^4$$

und die "Lösung"

$$x_2 = 1.00 \text{ (richtig)} \quad x_1 = 0.00 \text{ (falsch!)}$$

Vertauschen wir jedoch vor der Elimination die beiden Zeilen

$$\begin{aligned} &(\tilde{a}) \quad 1.00 x_1 + 1.00 x_2 = 2.00 \\ &(\tilde{b}) \quad 1.00e - 4 x_1 + 1.00 x_2 = 1.00 \end{aligned}$$

so folgt $\tilde{\ell}_{21} = 1.00 \cdot 10^{-4}$ und es ergibt sich das gestaffelte System

$$1.00 x_1 + 1.00 x_2 = 2.00$$

$$1.00 x_2 = 1.00$$

und die "richtige Lösung"

$$x_2 = 1.00 \quad x_1 = 1.00 \, .$$

□

Durch die Vertauschung der Zeilen hat man in obigem Beispiel erreicht, daß

$$|\tilde{\ell}_{21}| < 1 \text{ und } |\tilde{a}_{11}| \geq |\tilde{a}_{21}|$$

Das neue Pivotelement $\tilde{a}_{11}$ ist also das betragsmäßig größte Element in der ersten Spalte. Aus dieser Überlegung leitet man die sogenannte *Spaltenpivotstrategie* (engl.: *column pivoting* oder *partial pivoting*) ab. Bei jeder Gauß-Elimination wählt man diejenige Zeile als Pivotzeile, die das betragsmäßig größte Element in der Pivotspalte besitzt, formaler formuliert:

Algorithmus 1.4.2 Gauß-Elimination mit Spaltenpivotstrategie

a) Wähle im Eliminationsschritt $A^{(k)} \to A^{(k+1)}$ ein $p \in \{k, \ldots, n\}$, so daß

$$|a_{pk}^{(k)}| \geq |a_{jk}^{(k)}| \quad \text{für } j = k, \ldots, n$$

Die Zeile p soll die Pivotzeile werden.

b) Vertausche die Zeilen p und k

$$A^{(k)} \to \tilde{A}^{(k)} ,$$

$$\tilde{a}_{ij}^{(k)} = \begin{cases} a_{kj}^{(k)} & , \text{ falls } i = p \\ a_{pj}^{(k)} & , \text{ falls } i = k \\ a_{ij}^{(k)} & , \text{ sonst} \end{cases}$$

Nun gilt

$$|\tilde{\ell}_{ik}| = \left| \frac{\tilde{a}_{ik}^{(k)}}{\tilde{a}_{kk}^{(k)}} \right| = \left| \frac{\tilde{a}_{ik}^{(k)}}{a_{pk}^{(k)}} \right| \leq 1$$

c) Führe nun den nächsten Eliminationsschritt angewandt auf $\tilde{A}^{(k)}$ aus.

$$\tilde{A}^{(k)} \to A^{(k+1)} ,$$

□

Diese Strategie ist in den meisten Fällen erfolgreich. Es gibt jedoch auch Ausnahmen wie eine von Wilkinson untersuchte Matrix

$$A = \begin{bmatrix} 1 & & & 1 \\ -1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ -1 & \cdots & -1 & 1 \end{bmatrix}$$

zeigt, mit der wir uns um zweiten Kapitel beschäftigen werden.

Weiter fällt auf, daß sich die Pivotstrategie beliebig abändern läßt, indem man die einzelnen Zeilen mit verschiedenen Skalaren multipliziert. Diese Beobachtung führt uns zur Frage der *Skalierung.* Unter einer Zeilenskalierung verstehen wir eine Multiplikation von A mit einer Diagonalmatrix von links

$$A \to D_z A\ , \quad D_z \text{ Diagonalmatrix}$$

und analog unter einer Spaltenskalierung eine Multiplikation mit einer Diagonalmatrix von rechts

$$A \to A D_s\ , \quad D_s \text{ Diagonalmatrix}$$

(Wie wir bereits bei der Gauß-Elimination gesehen haben, läßt sich eine lineare Operation auf den Zeilen einer Matrix durch die Multiplikation mit einer geeigneten Matrix von links und entsprechend eine Operation auf den Spalten durch eine Matrizenmultiplikation von rechts beschreiben.)

Mathematisch gesprochen ändern wir mit der Skalierung die Länge der Basisvektoren des Bildraumes (Zeilenskalierung) bzw. Urbildraumes (Spaltenskalierung) der durch die Matrix A beschriebenen linearen Abbildung. Modelliert diese Abbildung einen physikalischen Sachverhalt, so kann man sich darunter die Änderung der Einheiten oder Umeichung vorstellen (z.B. von Å auf km). Damit die Lösung des linearen Gleichungssystems $Ax = b$ nicht von derartigen Wahlen abhängt, muß die Matrix A vom Algorithmus in geeigneter Weise skaliert werden:

$$A \to \tilde{A} := D_z A D_s$$

$$D_z = \operatorname{diag}(\sigma_1, \ldots, \sigma_n) := \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_1 \end{bmatrix}$$

$$D_s = \operatorname{diag}(\tau_1, \ldots, \tau_n)\ .$$

Recht vernünftig erscheinen die folgenden drei Strategien:

a) Eine erste Möglichkeit besteht in der *Äquilibrierung der Zeilen* von A bzgl. einer Vektornorm $\|\cdot\|$. Setzt man $D_s := I$ und

$$\sigma_i := \|A^i\|^{-1}\ , \quad i = 1, \ldots, n,$$

wobei A^i die i-te Zeile von A bezeichnet und vorausgesetzt wird, daß keine Zeile eine Nullzeile ist, so haben alle Zeilen von $\tilde{A}$ bzgl. $\|\cdot\|$ die Norm 1.

b) Analog kann man auch eine *Äquilibrierung der Spalten* durch $D_z := I$ und

$$\tau_j := \|A_j\|^{-1}\ , \quad j = 1, \ldots, n$$

erreichen, wobei A_j die j-te Spalte von A ist. In diesem Fall haben alle Spalten von $\tilde{A}$ die Norm 1.

c) Nach a) und b) ist es ganz natürlich zu fordern, daß sowohl die Zeilen als auch die Spalten von A jeweils untereinander dieselbe Norm haben. Zur Bestimmung der σ_i und τ_j hat man nun ein nichtlineares Gleichungssystem in $2n-2$ Unbekannten zu lösen. Wie man sich leicht vorstellen kann, erfordert dies bei weitem mehr Aufwand als die Lösung des ursprünglichen Problems. Wie wir im vierten Kapitel sehen werden, müssen dabei mehrere lineare Gleichungssysteme, jetzt mit $2n-2$ Unbekannten gelöst werden, für die sich natürlich wieder die Skalierungsfrage stellen würde.

Aufgrund dieser Lage wird die Skalierung in den meisten Programmen dem Benutzer überlassen.

Zur formalen Beschreibung der Dreieckszerlegung mit Spaltenpivotsuche verwenden wir Permutationsmatrizen P, welche bekanntermaßen die Permutationen der Einheitsvektoren $e_i = (0,\ldots,1,\ldots,0)$ beschreiben. Eine Permutation der Zeilen einer Matrix A läßt sich durch die Multiplikation von links mit einer solchen Permutationsmatrix ausdrücken

$$\text{Zeilenpermutation: } A \to PA$$

und wiederum vollkommen analog eine Permutation der Spalten durch die Multiplikation von rechts

$$\text{Spaltenpermutation: } A \to AP$$

Wie wir aus der linearen Algebra wissen, bilden die (n,n)-Permutationsmatrizen eine zur symmetrischen Gruppe S_n isomorphe Untergruppe der orthogonalen Gruppe $O(n)$, d.h. es gilt insbesondere

$$P^{-1} = P^T .$$

Jede Permutationsmatrix läßt sich als Produkt von Transpositionsmatrizen darstellen, also den Permutationen, die genau zwei Einheitsvektoren vertauschen. Die Determinante von P ist gerade das Vorzeichen der Permutation, also $+1$, falls P Produkt einer geraden Zahl von Transpositionen und -1 sonst.

$$\det P = \operatorname{sgn} P \in \{\pm 1\}$$

Der folgende Satz zeigt, daß die Dreieckszerlegung mit Spaltenpivotsuche theoretisch nur versagen kann, wenn die Matrix A singulär ist.

Satz 1.4.3 *Ist A eine nicht singuläre Matrix, dann existiert mindestens eine Permutationsmatrix P derart, daß eine Dreieckszerlegung*

$$PA = LR$$

möglich ist. Dabei kann P so gewählt werden, daß alle Elemente von L vom Betrag kleiner oder gleich 1 sind, d.h.

$$|L| \leq 1 .$$

Beweis: Wir führen den Algorithmus der LR-Zerlegung mit Spaltenpivotsuche aus. Da $\det A \neq 0$, gibt es eine Transpositionsmatrix P_1, so daß für $\tilde{A}^{(1)} := P_1 A$ gilt:

$$0 \neq |\tilde{a}_{11}^{(1)}| \geq |\tilde{a}_{i1}^{(1)}| , \quad i = 1,\ldots,n$$

Mit der oben eingeführten Notation folgt

$$A^{(2)} = \tilde{L}_1 \tilde{A}^{(1)} = \tilde{L}_1 P_1 A = \left[\begin{array}{c|c} a_{11}^{(2)} & * \cdots * \\ \hline 0 & \\ \vdots & B^{(2)} \\ 0 & \end{array}\right]$$

wobei $|\tilde{L}_1| \leq 1$. Da $a_{11}^{(2)} \neq 0$, folgt aus

$$0 \neq \det\ A = \det\ A^{(2)} = a_{11}^{(2)} \det\ B^{(2)},$$

daß $\det\ B^{(2)} \neq 0$. Wir erhalten so induktiv eine Zerlegung

$$R = A^{(n)} = \tilde{L}_{n-1} P_{n-1} \cdots \tilde{L}_1 P_1 A$$

wobei $|\tilde{L}_i| \leq 1$ und P_i evtl. zwei der Zeilen i bis n vertauscht. Man sieht leicht, daß es dann auch unipotente untere Dreiecksmatrizen $L_1, \ldots, L_{n-1}$ gibt mit $|L_i| \geq 1$ und

$$R = L_{n-1} \cdots L_1 P_{n-1} \cdots P_1 A\ .$$

Dazu müssen nur die Nichtdiagonalelemente der $\tilde{L}_i$ geeignet permutiert werden. Die gewünschte Zerlegung von A folgt daher mit

$$P := P_{n-1} \cdots P_1$$

$$L := L_1^{-1} \cdots L_{n-1}^{-1}\ .$$

□

Es sei noch angemerkt, daß wir die Determinante von A aus der Zerlegung $PA = LR$ wie in Satz 1.4.1 leicht mit der Formel

$$\det(A) = \operatorname{sgn}(P) \det(LR) = \operatorname{sgn}(P) \cdot r_{11} \cdots r_{nn}$$

berechnen können.

1.5 Cholesky-Verfahren für symmetrische, positiv definite Matrizen

Wir wollen nun die Gauß-Elimination auf eine eingeschränkte Klasse von Matrizen, die symmetrisch positiv definiten, anwenden. Es wird sich dabei herausstellen, daß die Dreieckszerlegung in diesem Fall stark vereinfacht werden kann.

Satz 1.5.1 *Sei A eine reelle symmetrische, positiv definite (n,n)-Matrix, d.h.*

$$\begin{array}{ll} \textit{(i)} & A^T = A \\ \textit{(ii)} & x^T A x > 0 \quad \forall x \neq 0 \end{array}$$

Dann ist A nicht singulär und es gilt:

a) $a_{ii} > 0$ *für* $i = 1, \ldots, n$.

b) $\max_{i=1,\ldots,n} a_{ii} = \max_{i,j=1,\ldots,n} |a_{ij}|$.

c) Bei der Gauß-Elimination ohne Pivotsuche ist jede Restmatrix wiederum positiv definit.

Insbesondere besagen b) und c), daß die Pivotsuche bei der LR-Zerlegung von A unnötig, ja sogar unsinnig ist, da sie evtl. die spezielle Struktur von A zerstört.

Beweis: zu a) Nach (ii) gilt für $x := e_i$, daß $a_{ii} = e_i^T A e_i > 0$.
zu b) Siehe Aufgabe ?.
zu c) Wir schreiben $A = A^{(1)}$ als

$$A^{(1)} = \left[\begin{array}{c|c} a_{11} & z^T \\ \hline z & B^{(1)} \end{array}\right]$$

wobei $z^T = (a_{12}, \ldots, a_{1n})$ und erhalten nach einem Eliminationsschritt

$$A^{(2)} = \left[\begin{array}{c|c} a_{11} & z^T \\ \hline 0 & B^{(2)} \end{array}\right]$$

Die Behauptung ist gerade, daß $B^{(2)}$ wieder symmetrisch positiv definit ist. Für die Elemente $a_{ij}^{(2)}$ $(i, j = 2, \ldots, n)$ von $B^{(2)}$ gilt

$$\begin{aligned} a_{ij}^{(2)} &= a_{ij}^{(1)} - \ell_{i1} a_{1j}^{(1)} \\ &= a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j} \\ &= a_{ij} - \frac{a_{1i} a_{1j}}{a_{11}}, \quad \text{da } a_{i1} = a_{1i} . \end{aligned}$$

Nun sind die Produkte $a_{1i}a_{1j}$ gerade die Elemente der Matrix zz^T, also

$$a_{1i}a_{1j} = (zz^T)_{ij}$$

(Eine Matrix der Gestalt xy^T für zwei Vektoren x, y nennen wir auch *dyadisches Produkt* von x und y. Matrizen dieser Art werden uns noch häufig begegnen).
Also gilt

$$B^{(2)} = B^{(1)} - \frac{1}{a_{11}}(zz^T) . \tag{1.5.1}$$

1. Die Symmetrie von $B^{(2)}$ folgt nun aus

$$\begin{aligned}(B^{(2)})^T &= (B^{(1)})^T - \frac{1}{a_{11}}(zz^T)^T \\ &= B^{(1)} - \frac{1}{a_{11}}(z^T)^T z^T \\ &= B^{(1)} - \frac{1}{a_{11}} zz^T \\ &= B^{(2)}\end{aligned}$$

2. Für die positive Definitheit haben wir zu zeigen, daß

$$y^T B^{(2)} y > 0 \quad \forall y = (y_2, \ldots, y_n) \in \mathbf{R}^n \ .$$

Mit (1.5.1) folgt:

$$\begin{aligned}y^T B^{(2)} y &= y^T B^{(1)} y - \frac{1}{a_{11}} y^T z z^T y \\ &= y^T B^{(1)} y - \frac{1}{a_{11}} (y^T z)^2\end{aligned}$$

Da A positiv definit ist, gilt

$$x^T A x > 0 \text{ für } x = (x_1, y^T) = (x_1, y_2, \ldots, y_n), \quad x_1 \in \mathbf{R}$$

und es folgt

$$\begin{aligned}0 < x^T A x &= (x_1, y^T) \begin{bmatrix} a_{11} & z^T \\ z & B^{(1)} \end{bmatrix} \begin{pmatrix} x_1 \\ y \end{pmatrix} \\ &= (x_1, y^T) \begin{pmatrix} a_{11} x_1 + z^T y \\ x_1 z + B^{(1)} x \end{pmatrix} \\ &= a_{11} x_1^2 + x_1 z^T y + x_1 (y^T z) + y^T B^{(1)} y \\ &= a_{11} (x_1 + \frac{z^T y}{a_{11}})^2 - \frac{(z^T y)^2}{a_{11}} + y^T B^{(1)} y \\ &= a_{11} (x_1 + \frac{z^T y}{a_{11}})^2 + y^T B^{(2)} y \ .\end{aligned}$$

Setzen wir speziell

$$x_1 := -\frac{z^T y}{a_{11}}$$

so folgt gerade $y^T B^{(2)} y > 0$. □

Zusammen mit dem Satz über die LR-Zerlegung können wir jetzt die *rationale Cholesky-Zerlegung* für symmetrische positiv definite Matrizen herleiten.

Satz 1.5.2 *Für jede symmetrische positiv definite Matrix A existiert eine eindeutig bestimmte Zerlegung der Form*

$$A = LDL^T,$$

wobei L eine unipotente untere Dreiecksmatrix und D eine positive Diagonalmatrix ist.

Beweis: Nach Satz 1 und Satz 1.4.1 c) gibt es eine eindeutig bestimmte Zerlegung

$$A = LR$$

mit einer unipotenten unteren Dreiecksmatrix L und einer oberen Dreiecksmatrix R, deren Diagonalelemente $a_{ii}^{(i)}$ alle positiv sind. Also gibt es eine positive Diagonalmatrix D und eine unipotente untere Dreiecksmatrix $\tilde{L}$, so daß

$$R = D\tilde{L}^T, \text{ also } A = LD\tilde{L}^T .$$

Da A symmetrisch ist, gilt

$$A = A^T = \tilde{L}DL^T = \tilde{L}\tilde{R},$$

wobei $\tilde{R} := DL^T$ eine obere Dreiecksmatrix ist. Aufgrund der Eindeutigkeit der LR-Zerlegung ist dann aber bereits $L = \tilde{L}$ und

$$A = \tilde{L}DL^T = LDL^T .$$

□

Korollar 1.5.3 *Da D positiv ist, existiert auch eine Zerlegung*

$$A = \bar{L}\bar{L}^T,$$

die Cholesky-Zerlegung, *wobei $\bar{L}$ die untere Dreiecksmatrix*

$$\begin{aligned} \bar{L} &:= L \cdot D^{1/2} \\ D^{1/2} &:= diag(\sqrt{d_1}, \ldots, \sqrt{d_n}) \text{ falls } D = diag(d_1, \ldots, d_n) \end{aligned}$$

Die Matrix $\bar{L} = (\ell_{ij})$ kann mit dem folgenden Algorithmus, dem *Cholesky-Verfahren*, berechnet werden:

Algorithmus 1.5.4 *Cholesky-Verfahren*

$$\begin{aligned} &\text{für } k = 1, \ldots, n: \quad \ell_{kk} := (a_{kk} - \textstyle\sum_{j=1}^{k-1} \ell_{kj}^2)^{1/2} \\ &\quad \text{für } i = k+1, \ldots, n: \quad \ell_{ik} = (a_{ik} - \textstyle\sum_{j=1}^{k-1} \ell_{ij}\ell_{kj})/\ell_{kk} \end{aligned}$$

Die Herleitung dieses Algorithmus ist nichts weiter als die elementweise Auswertung der Gleichung (1.5.7)

$$\begin{bmatrix} \ell_{11} & & \\ \vdots & \ddots & \\ \ell_{n1} & \ldots & \ell_{nn} \end{bmatrix} \begin{bmatrix} \ell_{11} & \ldots & \ell_{n1} \\ \ddots & & \vdots \\ & & \ell_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & \ldots & a_{nn} \\ \vdots & & \vdots \\ a_{n1} & \ldots & a_{nn} \end{bmatrix}$$

$$\begin{aligned} i = k: a_{kk} &= \ell_{k1}^2 + \cdots + \ell_{k,k-1}^2 + \ell_{kk}^2 \\ i > k: a_{ik} &= \ell_{i1}\ell_{k1} + \cdots + \ell_{i,k-1}\ell_{k,k-1} + \ell_{ik}\ell_{kk} . \end{aligned}$$

Die Auflösung der beiden Gleichungen nach ℓ_{kk} bzw. ℓ_{ik} liefert gerade die Formeln in (1.4.10). Die Raffinesse der Methode steckt in der Reihenfolge der Berechnung der Elemente von $\bar{L}$. Für den Rechenaufwand zählen wir

$$\sum_{k=1}^{n}(k-1+\sum_{i=k+1}^{n}k) \doteq \frac{1}{6}n^3$$

Multiplikationen und n Quadratwurzeln.

Bemerkungen:

1. Im Unterschied dazu kommt die rationale Cholesky-Zerlegung nur mit rationalen Operationen aus (daher der Name). Der Aufwand kann auch hier durch geschickte Programmierung auf $\frac{1}{6}n^3$ gehalten werden.

2. Ein Vorteil der rationalen Cholesky-Zerlegung besteht darin, daß fast singuläre Matrizen D erkannt werden können. Auch kann das Verfahren auf symmetrische indefinite Matrizen ($x^TAx \neq 0\ \forall x$) erweitert werden.

1.6 Übungen

Aufgabe 1.1 Geben sie eine vollbesetzte, nicht singuläre (3,3)-Matrix an, bei der das Gauß'sche Eliminationsverfahren versagt.

Aufgabe 1.2 Zeigen Sie, daß die LR-Zerlegung ohne Permutation eindeutig bestimmt ist.

Aufgabe 1.3 (Gauß-Elimination bei Spezialstruktur)
Sei $A = (a_{ij})$ eine Matrix mit folgenden Eigenschaften:

(i) $\sum_{i=1}^{n} a_{ij} = 0$ für $j = 1, \ldots, n$

(ii) $a_{ii} < 0$ und $a_{ij} \geq 0$ für $i = 1, \ldots, n$ und $j \neq i$

Wie in der Vorlesung bezeichnen wir mit $A = A^{(1)}, A^{(2)}, \ldots, A^{(n)}$ die bei der Gauß'schen Elimination entstehenden Matrizen. Zeigen Sie:

(a) $|a_{11}| \geq |a_{i1}|$ für $i = 2, \ldots, n$

(b) $\sum_{i=2}^{n} a_{ij}^{(2)} = 0$ für $j = 2, \ldots, n$

(c) $a_{ii}^{(1)} \leq a_{ii}^{(2)} \leq 0$ für $i = 2, \ldots, n$

(d) $a_{ij}^{(2)} \geq a_{ij}^{(1)} \geq 0$ für $i, j = 2, \ldots, n$ und $j \neq i$

(e) Falls die bei den ersten $n-2$ Gauß-Eliminationen sukzessiv entstehenden Diagonalelemente alle verschieden von Null sind, d. h. $a_{ii}^{(i)} < 0$ für $i = 1, \ldots, n-1$, so gilt $a_{nn}^{(n)} = 0$.

Aufgabe 1.4 Programmieren Sie das Gauß'sche Eliminationsverfahren. Das Programm soll die Daten A und b aus einer Datei einlesen und mit folgenden Beispielen getestet werden:

(a) mit der Matrix aus Aufgabe 1,

(b) mit $n = 1$, $A = 25$ und $b = 4$,

(c) mit $a_{ij} = i^{j-1}$ und $b_i = i$ für $n = 7$, 15 und 50.

Vergleiche jeweils die berechneten mit den exakten Lösungen.

Aufgabe 1.5 Die Gauß-Zerlegung mit Spaltenpivotsuche liefere angewandt auf die Matrix A die Zerlegung $PA = LR$, wobei P die sich im Lauf der Elimination ergebende Permutationsmatrix ist. Zeigen Sie:

(a) Die Gauß-Elimination mit Spaltenpivotsuche ist invariant gegen

(i) Permutationen der Zeilen von A (triviale Ausnahme: mehrere betragsgleiche Elemente pro Spalte)

(ii) Multiplikation der Matrix mit einer Zahl $\sigma \neq 0$, $A \longrightarrow \sigma A$.

(b) Ist D eine Diagonalmatrix, so liefert die Gauß-Elimination mit Spaltenpivotsuche angewandt auf $\bar{A} := AD$ die Zerlegung $P\bar{A} = L\bar{R}$ mit $\bar{R} = RD$.

Aufgabe 1.6 (a) Zeigen Sie, daß die normierten unteren Dreiecksmatrizen

$$\{L = (l_{ij}) \mid l_{ii} = 1 \text{ für } i = 1, \ldots, n \text{ und } l_{ij} = 0 \text{ für } 1 \leq i < j \leq n\}$$

eine Untergruppe der speziellen linearen Gruppe

$$SL(n) = \{A \in \mathrm{Mat}(n, \mathbf{R}) \mid \det(A) = 1\}$$

bilden.

(b) Sei A eine symmetrische positiv definite Matrix. Zeigen Sie, daß dann

$$\max_{i=1,\ldots,n} a_{ii} = \max_{i,j=1,\ldots,n} |a_{ij}|.$$

Aufgabe 1.7 Ein Problem aus der Astrophysik führt auf ein System von $(n+1)$ linearen Gleichungen in n Unbekannten der Form

$$\begin{pmatrix} & & \\ & A & \\ & & \\ 1 & \cdots & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix},$$

wobei A die Matrix aus Aufgabe 1.3 ist. Zur Lösung dieses Systems wenden wir auf die Matrix A eine Gauß-Elimination ohne Pivotsuche mit folgenden zwei Zusatzregeln an, wobei wir die entstehenden Matrizen wieder mit $A = A^{(1)}, \ldots, A^{(n-1)}$ und die relative Maschinengenauigkeit mit eps bezeichnen.

(a) Falls im Laufe des Algorithmus $|a_{kk}^{(k)}| \leq |a_{kk}|$ eps für ein $k < n$ zum ersten Mal auftritt, so vertausche die Zeilen n und k und die Spalten n und k.

(b) Falls $|a_{kk}^{(k)}| \leq |a_{kk}|$ eps für ein weiteres $k < n$, so breche den Algorithmus ab.

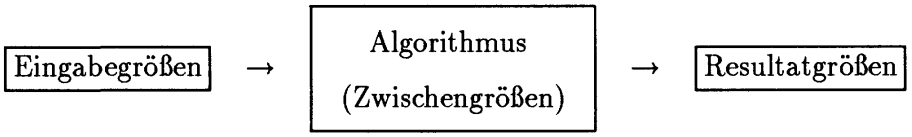
Zeigen Sie, daß der Algorithmus, falls er nicht in (b) abbricht, nach $n-1$ Eliminationsschritten eine Zerlegung von A in $PAP = LR$ liefert, wobei P eine Permutation und $R = A^{(1)}$ eine obere Dreiecksmatrix mit $r_{nn} = 0$, $r_{ii} < 0$ für $i = 1, \ldots, n-1$ und $r_{ij} \geq 0$ für $j > i$ ist. Zeigen Sie weiter, daß das System in diesem Fall eine eindeutige Lösung x besitzt und alle Komponenten von x nicht negativ sind. Geben Sie eine einfache Vorschrift zur Berechnung von x an.

Aufgabe 1.8 Programmieren Sie den in Aufgabe 1.7 entwickelten Algorithmus zur Lösung des speziellen Gleichungssystems und testen Sie das Programm an zwei selbstgewählten Beispielen der Dimension $n = 5$ und $n = 7$, sowie mit der Matrix

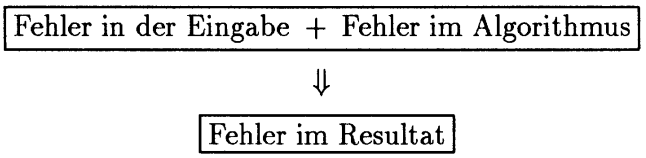
$$\begin{pmatrix} -2 & 2 & 0 & 0 \\ 2 & -4 & 1 & 1 \\ 0 & 2 & -1 & 1 \\ 0 & 0 & 0 & -2 \end{pmatrix}$$

2 Fehleranalyse

In diesem Kapitel wollen wir uns damit beschäftigen, wie sich Fehler bei der rechnerischen Lösung eines Problems auswirken. Dabei müssen wir uns zunächst klarmachen, welche Arten von Fehlern auftreten können. Bei der rechnerischen Lösung eines Problems geht es stets darum, aus gewissen Eingabegrößen mit Hilfe eines Algorithmus, der evtl. noch Zwischengrößen produziert, eine Anzahl von Resultaten zu berechnen.



Aus den Eingabefehlern und den Fehlern im Algorithmus resultieren Fehler im Ergebnis:



Bei der Fehleranalyse ist es insbesondere wichtig, zwischen dem Einfluß von Eingabefehlern und dem von Fehlern im Algorithmus auf den Fehler im Ergebnis zu unterscheiden. Dazu werden wir im folgenden die Begriffe *Kondition eines Problems* bzw. *Stabilität eines Algorithmus* einführen.

2.1 Fehlerquellen

2.1.1 Eingabefehler

a) Meßfehler

Bei Fehlern der Eingabegrößen denken wir zunächst an *Meßfehler*, falls die Eingabegrößen z. B. experimentelle Daten sind. Solche Eingabegrößen $\bar{z}_i$ werden meist mit Angaben über den absoluten Fehler $\delta\bar{z}_i$, den *Toleranzen*, angegeben, so daß für den Abstand zu den "wahren" Werten z_i die Abschätzung

$$|\bar{z}_i - z_i| \leq \delta z_i$$

gilt. Die relative Genauigkeit $|\delta z_i / z_i|$ liegt dabei gewöhnlich bei 10^{-2} bis 10^{-3}.

a) Rundungsfehler

Aber auch wenn die Eingabegrößen exakt gedacht werden können, treten durch die Darstellung einer nicht ganzzahligen Zahl im Rechner Eingabefehler auf. Bei der heute üblichen (*normalisierten*) *Gleitkommadarstellung* wird eine Zahl $\bar{z}$ vom Typ "real" dargestellt als $\bar{z} = a \cdot d^e$, wobei die *Basis* d eine Zweierpotenz ist (in der Regel

$d \in \{2,8,16\}$), der *Exponent* e eine ganze Zahl mit einer vorgegebenen maximalen (Binär-) Stellenzahl, und die *Mantisse* a entweder 0 oder eine mit einer endlichen Anzahl von Stellen, der sogenannten *Mantissenlänge*, dargestellte Zahl mit

$$\frac{1}{d} \leq |a| < 1$$

Beispiel 2.1.1 (die alt-ehrwürdige IBM 360)

Basis $d = 2^4$, Exponentenlänge $h = 2$, Mantissenlänge $\ell = 6$, Vorzeichen $V \in \{0,1\}$, $A_i \in \{0,\ldots,15\}$, $e \in \{-64,\ldots,63\}$

V	e	A_1	A_2	...	A_6
1 Bit	7 Bits	4 Bits	4 Bits	...	4 Bits

Zahlenbereich:

$$\frac{1}{16} \cdot 16^{-64} \leq |\bar{z}| \leq (1 - \frac{1}{16^6}) \cdot 16^{63}$$

Der Bereich, den der Exponent e überstreichen kann, bestimmt die größte und die kleinste Zahl, die sich auf der Maschine (damit ist hier der Rechner zusammen mit dem benutzten Compiler gemeint) darstellen läßt, die Mantissenlänge ℓ die *relative Genauigkeit.* Jede Zahl z läßt sich auf dem Rechner durch eine Zahl $\bar{z}$ repräsentieren, so daß

$$\frac{|\bar{z} - z|}{|z|} \leq \text{ eps} := \frac{1}{2} d^{1-\ell} \text{ (bei Rundung)}$$

Wir nennen eps die *relative Maschinengenauigkeit*, in der Literatur auch mit **u** für "unit roundoff" bezeichnet. Sie liegt in der Regel bei eps $\approx 10^{-7}$ bei einfacher Genauigkeit (single precision in FORTRAN, float in *C*). Betrachten wir nun als Beispiel die mathematisch exakt zu denkende Zahl

$$\pi = 3.141592653589...$$

π kann als transzendente Zahl nicht mit einer endlichen Mantissenlänge dargestellt werden und ist daher auf jedem Rechner eine fehlerhafte Eingabegröße !

$$\pi \rightarrow \bar{\pi} = 3.141593$$

$$\frac{|\bar{\pi} - \pi|}{|\pi|} \leq \text{ eps}$$

Allgemein sind in der Regel alle nicht ganzzahligen Eingabegrößen mit Eingabefehlern durch die Rundung auf eine Maschinenzahl behaftet. Es ist insofern unsinnig, von einer "exakten" Eingabe auszugehen. Diese Einsicht wird unsere Fehleranalyse maßgeblich beeinflussen.

2.1.2 Fehler im Algorithmus

Bei den Fehler, die bei der Ausführung des Algorithmus entstehen, unterscheiden wir zwischen drei Arten:

a) **Rundungsfehler** bei der Ausführung der Elementaroperationen +, -, *, /

b) **Approximationsfehler**, z.B. bei der Berechnung von $\sin x$ durch eine abgebrochene Reihenentwicklung.

c) **Diskretisierungsfehler**, z.B. bei der numerischen Integration (Quadratur, s. Kapitel 6) oder der numerischen Lösung von Differentialgleichungen.

Diskretisierungsfehler treten immer dann auf, wenn ein ursprünglich unendlich dimensionales Problem auf ein rechnerisch zugängliches endliches Problem projiziert wird. Diese Fehler sowie manchmal auch Approximationsfehler sind wesentlich schwerer abzuschätzen als Rundungsfehler, auf die wir uns in diesem Kapitel beschränken.

Wir bezeichnen mit $\mathrm{fl}(z)$ die durch Rundung der Zahl z zugeordnete Gleitkommazahl $\bar{z} = \mathrm{fl}(z)$, wobei "fl" für "floating point" steht. Dann läßt sich der Rundungsfehler bei der Ausführung einer Elementoperation $\circ \in \{+, -, *, /\}$ in Gleitkommaarithmetik für zwei Maschinenzahlen $\bar{a}$ und $\bar{b}$ abschätzen durch

$$\mathrm{fl}(\bar{a} \circ \bar{b}) = (\bar{a} \circ \bar{b})(1 + \varepsilon) \text{ mit } |\varepsilon| \leq \text{ eps}$$

2.2 Kondition eines Problems

Wir gehen aus von der Frage:

> "Wie wirken sich Störungen der Eingabegrößen auf das Resultat *unabhängig* vom gewählten Algorithmus aus" ?

Wenden wir uns nochmals der Eingabe von π in einen Rechner zu, so fällt auf, daß die vormals "exakte" Zahl π nach der Eingabe in einen Rechner und damit der Rundung auf $\bar{\pi}$ *logisch ununterscheidbar* ist von allen Zahlen z, die durch Rundung auf dieselbe Maschinenzahl $\bar{\pi}$ führen, also alle z mit

$$\frac{|z - \bar{\pi}|}{|z|} \leq \text{ eps}$$

Statt der "exakten" Eingabe π sollten wir also besser eine *Eingabemenge*

$$E := \{z \in \mathbf{R} \mid \frac{|z - \bar{\pi}|}{|z|} \leq \text{ eps}\}$$

betrachten. Allgemeiner wäre statt einer eingegebenen Maschinenzahl $\bar{e}$ die Eingabemenge

$$\begin{aligned} E &:= \{e \mid \frac{|e - \bar{e}|}{|e|} \leq \text{ eps}\} \\ &= \{e \mid e = \bar{e}(1 + \varepsilon), \ |\varepsilon| \leq \text{ eps}\} \end{aligned}$$

die eigentliche Eingabegröße. Dabei haben wir bisher nur die durch Rundung entstehenden Eingabefehler berücksichtigt. Wüßten wir z.B., daß die Eingabegröße $\bar{z}$ mit einem relativen Meßfehler von $\varepsilon = 10^{-2} >$ eps behaftet ist, so ist die entscheidende Eingabegröße die Menge

$$E := \{z \mid \frac{|z - \bar{z}|}{|z|} \leq 10^{-2}\}$$

Alle $z \in E$ sind wegen der angegebenen relativen Toleranz ununterscheidbar.

Formulieren wir unser Problem mathematisch als eine Abbildung f,

$$f : e \rightarrow r = f(e)$$

die einer Eingabe e ein Resultat $r = f(e)$ zuordnet, so überführt die gleiche Abbildung die Eingabemenge E in eine *Resultatmenge* $R = f(E)$. Dabei können wir uns die Funktion f als "exakten", d.h. fehlerfreien Algorithmus denken (wir wollen ja gerade die Störungen des Resultats unabhängig vom Algorithmus untersuchen.)

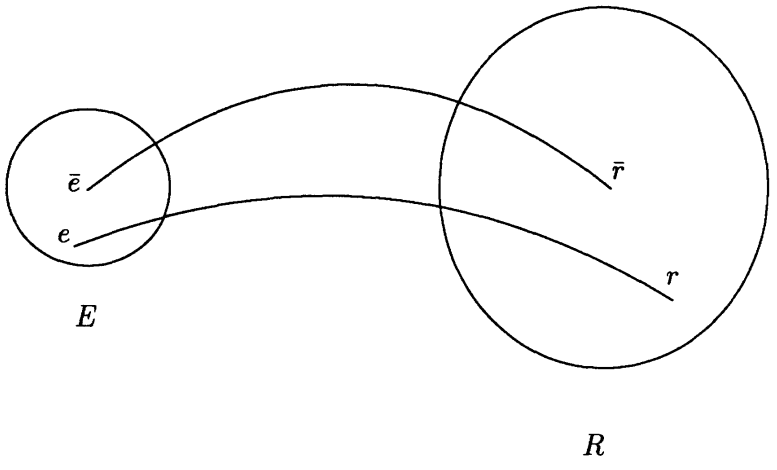


Abbildung 2.1: Eingabe- und Resultatmenge

Bei der Analyse von Eingabefehlern ist es daher sinnvoll, anstelle der *punktweisen* Abbildung $f : e \rightarrow r = f(e)$ die *Mengenabbildung* $f : E \rightarrow R = f(E)$ zu untersuchen, wobei die Eingabemenge E jeweils problemspezifisch zu wählen ist, wie wir in den beiden einfachen Beispielen gesehen haben. Die Auswirkung von Störungen der Eingabegrößen auf das Resultat lassen sich nun an dem Verhältnis der Eingabemenge zur Resultatmenge ablesen. Eine Charakterisierung des Verhältnisses von E und R nennen wir *Kondition des durch f beschriebenen Problems.* Die mathematische Konkretisierung

dieses Begriffes hängt dabei vom jeweiligen Problem ab. Können wir für die Eingabe- und Resultatmengen auf vernünftige Weise "Maße" $[E]$ und $[R]$ definieren (nicht im maßtheoretischen Sinn zu verstehen), die die "Größe" der Mengen widerspiegeln, so charakterisiert der Quotient

$$\kappa := \frac{[R]}{[E]}$$

die Kondition des Problems. Beispiele für solche "Maße" für Mengen $E, R \subset \mathbf{R}$ sind:

a) der *relative Fehler*

$$\begin{aligned} [E]_{\text{rel}} &:= \sup_{e \in E} \frac{|e - \bar{e}|}{|e|} \\ [R]_{\text{rel}} &:= \sup_{r \in R} \frac{|r - \bar{r}|}{|r|} = \sup_{e \in E} \frac{|f(e) - f(\bar{e})|}{|f(e)|} \end{aligned}$$

b) der *absolute Fehler*

$$\begin{aligned} [E]_{\text{abs}} &:= \sup_{e \in E} |e - \bar{e}| \\ [R]_{\text{abs}} &:= \sup_{e \in R} |r - \bar{r}| = \sup_{e \in E} |f(e) - f(\bar{e})| \end{aligned}$$

die auf die *relative Kondition*

$$\kappa_{\text{rel}} := \frac{[E]_{\text{rel}}}{[R]_{\text{rel}}}$$

bzw. die *absolute Kondition*

$$\kappa_{\text{abs}} := \frac{[E]_{\text{abs}}}{[R]_{\text{abs}}})$$

führen. Wie man sieht, ist das relative Fehlerkonzept nur durchhaltbar, falls weder E noch R die 0 enthalten. Ferner beachte man, daß wir in a) und b) bisher nur Teilmengen der rellen Zahlen betrachtet haben. Für vektorwertige e und r sind die Beträge durch Normen zu ersetzen, von deren Wahl κ dann natürlich abhängt. Wir können nun definieren, wann ein Problem gut und wann es schlecht konditioniert zu nennen ist. Der Einfachheit halber sei nur die relative Kondition gewählt.

Definition 2.2.1 *Ein Problem heißt bezüglich des relativen Fehlers*

a) gut konditioniert *(well-conditioned, well-posed), falls*

$$\kappa_{\text{rel}} \approx 1$$

b) schlecht konditioniert *(ill-conditioned), falls*

$$\kappa_{\text{rel}} \gg 1$$

c) unsachgemäß gestellt *(ill-posed), falls*

$$\kappa_{\text{rel}} = \infty.$$

Bemerkung 2.2.2 Falls $\kappa_{\text{rel}} \geq \dfrac{1}{[E]_{\text{rel}}}$, so ist eventuell

$$|\frac{f(e) - f(\bar{e})}{f(e)}| \geq 1$$

für ein $e \in E$, und damit $f(\bar{e})$ zu 100% falsch !

Wir haben die Kondition oben nur für eine feste Eingabemenge E definiert. Läßt sich eine solche Eingabemenge nicht sinnvoll angeben, so analysiert man häufig das Verhalten der Resultatmenge R in Abhängigkeit von der Eingabemenge E für $[E] \to 0$. Dazu betrachtet man die *asymptotische Kondition*

$$\bar{\kappa} := \limsup_{[E]\to 0} \frac{[R]}{[E]} = \lim_{\varepsilon \to 0} \sup_{[E]=\varepsilon} \frac{[R]}{[E]} \tag{2.2.1}$$

Wir werden darauf bei der Analyse der Kondition eines linearen Gleichungssystems zurückkommen.

Der Begriff der Kondition soll nun an einigen Beispielen erläutert werden:

2.2.1 Berechnung des Schnittpunktes zweier Geraden

Als erstes betrachten wir das geometrische Problem der Bestimmung des Schnittpunktes r zweier Geraden g, h in der Ebene

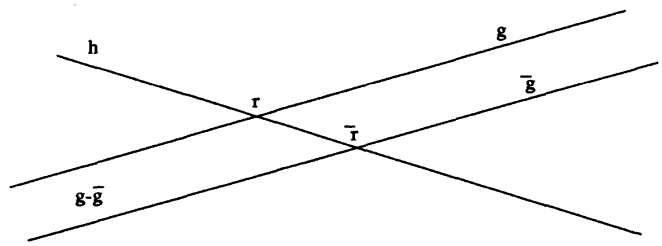


Abbildung 2.2: Schnittpunktes r zweier Geraden g, h.

Bei der zeichnerischen Lösung dieses Problems ist es nicht möglich, die Geraden g und h exakt darzustellen. Die Frage ist nun, wie stark der konstruierte Schnittpunkt r von den "Zeichenfehlern" abhängt. Der Einfachheit halber gehen wir davon aus, daß h "exakt" vorliege (oder der Fehler gegenüber dem von g vernachlässigbar sei) und $\bar{g}$ nur um eine

Parallelverschiebung von g abweiche, d.h. $g \parallel \bar{g}$. Dann können wir den Fehler durch den Normalabstand $|g - \bar{g}|$ messen. Die Eingabemenge besteht aus allen Geraden g, deren Normalabstand von $\bar{g}$ unterhalb der Zeichengenauigkeit ε liegt

$$E = \{g \mid |g - \bar{g}| \leq \varepsilon\}$$

Ein vernünftiges (absolutes) Maß ist gerade dieser maximale Abstand

$$[E]_{\text{abs}} = \sup_{g \in E} |g - \bar{g}| = \varepsilon$$

Für die Resultatmenge

$$R = \{r \mid r \text{ ist Schnittpunkt von } g \text{ und } h, \quad g \in E\}$$

ist das geeignete Fehlermaß

$$[R]_{\text{abs}} = \sup_{r \in R} |r - \bar{r}|$$

Bei der Konditionsanalyse springen im wesentlichen zwei Grenzfälle ins Auge.

a) *g und h stehen senkrecht zueinander, $g \perp h$.*

$$\Rightarrow \kappa_{\text{abs}} = \frac{[E]_{\text{abs}}}{[R]_{\text{abs}}} = 1$$

Falls also $g \perp h$ oder $\sphericalangle(g,h) \approx \pi/2$, so ist das Problem gut konditioniert.

b) *g und h sind nahezu parallel*

$$\sphericalangle(g,h) \to 0 \Rightarrow \kappa_{\text{abs}} \to \infty$$

Für kleine Winkel $\sphericalangle(g,h)$ ist die Schnittpunktbestimmung schlecht konditioniert. Geometrisch gesprochen liegt ein sogenannter *schleifender Schnitt* vor.

Abbildung 2.3: Schleifender Schnitt.

2.2.2 Subtraktion und Addition

Zu berechnen sei zunächst die Differenz

$$r = a - b, \quad a > b > 0$$

zweier reeller Zahlen a und b. Durch Rundung erhalten wir die Maschinenzahlen

$$\bar{a} = \mathrm{fl}(a) = a(1 + \varepsilon_a), \quad |\varepsilon_a| \leq \text{ eps}$$

$$\bar{b} = \mathrm{fl}(b) = b(1 + \varepsilon_b), \quad |\varepsilon_b| \leq \text{ eps}$$

Nach dem oben gesagten ist hier die Eingabemenge

$$E := \{(a, b) \mid \ a = \frac{\bar{a}}{1 + \varepsilon_a}, \quad b = \frac{\bar{b}}{1 + \varepsilon_b}, \quad |\varepsilon_a|, \ |\varepsilon_b| \leq \text{ eps}\}$$

zu betrachten mit

$$[E]_{\text{rel}} = \sup_{(a,b) \in E} \left\{ \frac{|a - \bar{a}|}{|a|}, \frac{|b - \bar{b}|}{|b|} \right\} = \text{ eps}$$

Nun gilt für $(a, b) \in E$ und $r := a - b$, daß

$$\begin{aligned} \bar{r} &:= \bar{a} - \bar{b} \\ &= a(1 + \varepsilon_a) - b(1 + \varepsilon_b) \\ &= (a - b)(1 + \frac{a}{a - b}\varepsilon_a - \frac{b}{a - b}\varepsilon_b) \\ &= r(1 + \frac{a}{a - b}\varepsilon_a - \frac{b}{a - b}\varepsilon_b) \end{aligned}$$

also

$$\frac{|r-\bar{r}|}{|r|} = \left| \frac{a}{a-b}\varepsilon_a - \frac{b}{a-b}\varepsilon_b \right|$$

Für $[R]_{\text{rel}}$ erhalten wir daher

$$\begin{aligned} [R]_{\text{rel}} &:= \sup_{r \in R} \frac{|r-\bar{r}|}{|r|} \\ &= \sup_{|\varepsilon_a|,|\varepsilon_b| \leq \text{eps}} \left| \frac{a}{a-b}\varepsilon_a - \frac{b}{a-b}\varepsilon_b \right| \\ &= \frac{a+b}{a-b} \text{ eps}. \end{aligned}$$

Für beliebige $a, b \in \mathbf{R}$ folgt daraus für die relative Kondition der Subtraktion $a-b$

$$\kappa_{\text{rel}} = \frac{[R]_{\text{rel}}}{[E]_{\text{rel}}} = \frac{|a|+|b|}{|a-b|}.$$

Im Fall der *Addition* (hier: a und b haben verschiedene Vorzeichen) ergibt sich sofort

$$\kappa_{\text{rel}} = 1.$$

Bei der *Subtraktion* hingegen gilt:

$$|a-b| \ll |a|+|b| \iff \kappa_{\text{rel}} \gg 1$$

Dies ist gerade die *Auslöschung* (engl.: *cancellation of leading digits*) bei der Subtraktion nahezu identischer Zahlen, wie sie in dem folgenden einfachen Beispiel mit $\text{eps} = 10^{-7}$ auftritt:

$$\begin{aligned} \bar{a} &= 0.\,1234\,67\,* \quad \leftarrow \text{Störung an 7. Stelle} \\ \bar{b} &= 0.\,1234\,56\,* \\ \bar{r} = \bar{a} - \bar{b} &= 0.\underbrace{0000}\,11\,*\,\underbrace{000} \quad \leftarrow \text{Störung an 3. Stelle} \\ &\quad \text{führende Nullen} \quad \text{nachgeschobene Nullen} \end{aligned}$$

Ein Fehler in der 7. Dezimalstelle führt im Ergebnis zu einem Fehler in der 3. Dezimalstelle, d.h. $\kappa_{\text{rel}} \approx 10^4$. Unsere Konditionsanalyse zeigt, daß die Subtraktion im Fall der Auslöschung schlecht konditioniert ist. Wir ziehen daraus die (nur tautologisch klingende) Regel:

Vermeide vermeidbare Subtraktionen

Bemerkung 2.2.3 Man beachte, daß man die Auslöschung dem vom Rechner ausgegebenen dezimalen Ergebnis nicht ansehen kann. Die nachgeschobenen Nullen sind Nullen in der *binären* Darstellung und gehen bei der Umwandlung ins Dezimalsystem verloren.

Beispiel 2.2.4 Beispiel für eine vermeidbare Subtraktion:

$$\begin{aligned}\frac{1-\cos(x)}{x} &= \frac{1}{x}\left(1-[1-\frac{x^2}{2}+\frac{x^4}{24}\pm\cdots]\right)\\ &= \frac{x}{2}\left(1-\frac{x^2}{12}\pm\cdots\right)\end{aligned}$$

Für $x = 10^4$ ist $x^2/12 < 10^{-9}$ und daher $x/2$ eine auf acht Dezimalziffern genaue Näherung für $(1-\cos\ x)/x$.

2.2.3 Berechnung des Skalarproduktes

Wir wollen das Skalarprodukt $< a, b >$ zweier Vektoren $a, b \in \mathbf{R}^n$ berechnen.

$$r = f(a,b) = s_n = a^T b = \sum_{i=1}^{n} a_i b_i$$

$$\begin{aligned}&\text{Eingabedaten} \quad : \quad a_1, \ldots, a_n, b_1, \ldots, b_n\\ &\text{Resultat} \qquad\quad : \quad s_n\end{aligned}$$

Gehen wir wieder von Störungen der Eingabegrößen aus, wie sie bei der Rundung

$$\begin{aligned}a_i \to \bar{a}_i &= a_i(1+\varepsilon_i) \quad , \quad |\varepsilon_i| \le \text{ eps}\\ b_i \to \bar{b}_i &= b_i(1+\delta_i) \quad , \quad |\delta_i| \le \text{ eps}\end{aligned}$$

auftreten, so kommen wir zu der Eingabemenge

$$E := \{(a,b) \mid \ a_i = \frac{\bar{a}_i}{1+\varepsilon_i}, b_i = \frac{\bar{b}_i}{1+\delta_i} \ \text{ mit } \ |\varepsilon_i|,\ |\delta_i| \le \text{ eps}\}$$

Die Fehler von a_i und b_i werden im Skalarprodukt $s_n = \sum_{i=1}^{n} a_i b_i$ jeweils miteinander verkoppelt,

$$a_i b_i \to \bar{a}_i \bar{b}_i = a_i b_i (1+\varepsilon_i)(1+\delta_i)$$

so daß es keine Rolle spielt, wie der Fehler des Produktes $a_i b_i$ auf die beiden Faktoren a_i und b_i verteilt ist. Ein vernünftiges Fehlermaß für Eingabegrößen ist daher

$$[E]_{\text{rel}} := \sup_{(a,b)\in E} \max_i \frac{|a_i b_i - \bar{a}_i \bar{b}_i|}{|a_i b_i|}$$

Da

$$\begin{aligned}\frac{|a_i b_i - \bar{a}_i \bar{b}_i|}{|a_i b_i|} &= \frac{|a_i b_i - a_i b_i (1+\varepsilon_i)(1+\delta_i)|}{|a_i b_i|}\\ &= |\varepsilon_i + \delta_i + \varepsilon_i \delta_i|\\ &\doteq |\varepsilon_i + \delta_i|\end{aligned}$$

gilt

$$[E]_{\text{rel}} \doteq 2 \text{ eps}.$$

Analog zu den Berechnungen bei der Subtraktion gilt ferner für $\bar{r} = \bar{a}^T \bar{b}$:

$$\begin{aligned} \bar{r} &= \sum_{i=1}^{n} \bar{a}_i \bar{b}_i \\ &= \sum_{i=1}^{n} a_i b_i (1+\varepsilon_i)(1+\delta_i) \\ &= (\sum_{i=1}^{n} a_i b_i) \left(1 + \sum_{i=1}^{n} \frac{a_i b_i (\varepsilon_i + \delta_i + \varepsilon_i \delta_i)}{\sum_{j=1}^{n} a_j b_j}\right) \end{aligned}$$

also

$$\frac{r - \bar{r}}{r} = \frac{\sum_{i=1}^{n} \bar{a}_i \bar{b}_i (\varepsilon_i + \delta_i + \varepsilon_i \delta_i)}{\sum_{i=1}^{n} \bar{a}_i \bar{b}_i}$$

und daher

$$\begin{aligned} [R]_{\text{rel}} &= \sup_{r \in R} \frac{|r - \bar{r}|}{|r|} \\ &= 2 \text{ eps} \frac{\sum_{i=1}^{n} |\bar{a}_i \bar{b}_i|}{|\sum_{i=1}^{n} \bar{a}_i \bar{b}_i|} \\ &= 2 \text{ eps} \frac{|\bar{a}|^T |\bar{b}|}{|\bar{a}^T \bar{b}|} \end{aligned}$$

Für die relative Kondition des Skalarproduktes erhalten wir somit

$$\kappa_{\text{rel}} = \frac{[R]_{\text{rel}}}{[E]_{\text{rel}}} = \frac{|\bar{a}|^T |\bar{b}|}{|\bar{a}^T \bar{b}|}$$

Beispiel 2.2.5 Als Anwendung der Konditionsanalyse des Skalarproduktes betrachten wir die Berechnung von e^x durch die abgebrochene Taylor-Reihe

$$e^x \approx s_N := 1 + x + \frac{x^2}{2!} + \ldots + \frac{x^N}{N!} = \sum_{i=0}^{N} a_i b_i = a^T b$$

mit

$$\begin{aligned} a_0 &:= 1 \quad , \quad a_{i+1} := x a_i \\ b_0 &:= 1 \quad , \quad b_{i+1} := b_i / (i+1) \end{aligned}$$

Dabei geht es uns hier nicht um den Approximationsfehler $|e^x - s_N|$, sondern nur um den Fehler bei der Berechnung der Summe $s_N = \sum_{i=0}^{N} a_i b_i$.

a) Falls $x > 0$, so gilt $a_i b_i > 0$ und daher

$$\kappa_{\text{rel}} = \frac{|a|^T |b|}{|a^T b|} = 1 \ .$$

b) Falls jedoch $x < 0$ (alternierende Reihe), so ist

$$\begin{aligned} a^T b &= s_N \doteq e^x = e^{-|x|} \\ |a^T||b| &\doteq e^{-x} = e^{|x|} \end{aligned}$$

also

$$\kappa_{\text{rel}} = e^{2|x|} \geq 1$$

Die abgebrochene Taylorreihe ist daher (von den Approximationseigenschaften ganz abgesehen) allenfalls für "kleine" $x < 0$ ein geeigneter Algorithmus. da für $x < 0$ die Auswertung der Summe s_N schlecht konditioniert ist. Tatsächlich wird die Exponentialfunktion heute mit einer Modifikation des sogenannten *Brigg'schen Algorithmus* berechnet.

2.2.4 Nullstellenbestimmung von nichtlinearen Funktionen

Gegeben sei eine Funktion $f : \mathbf{R} \to \mathbf{R}$ und wir fragen nach der Kondition der Bestimmung einer Nullstelle x^* von f, d.h. $f(x^*) = 0$. Wir setzen voraus, daß f in einer Umgebung U von x^* stetig differenzierbar und injektiv ist, so daß die Umkehrfunktion

$$f^{-1} : (-\delta, \delta) \to U, \quad \text{für ein } \delta > 0$$

in einer Nullumgebung $(-\delta, \delta)$ existiert.

Betrachten wir einen Auswertungsfehler ε von f und $\bar{x}^* := f^{-1}(\varepsilon)$, so folgt

$$\begin{aligned} |\bar{x}^* - x^*| &= |f^{-1}(\varepsilon) - f^{-1}(0)| \\ &\doteq \left| \frac{df^{-1}(\varepsilon)}{d\varepsilon}|_{\varepsilon=0} \right| \cdot \varepsilon \\ &= \frac{1}{|f'(x^*)|} \cdot \varepsilon \end{aligned}$$

Für die absolute Kondition bei der Nullstellenbestimmung ergibt sich daraus

$$\kappa_{\text{abs}} = \frac{|\bar{x}^* - x^*|}{|\varepsilon|} = \frac{1}{|f'(x^*)|}$$

Bezüglich des absoluten Fehlers ist daher insbesondere die Bestimmung *mehrfacher* oder *nahe zusammenliegender* Nullstellen schlecht konditioniert. Anschaulich deckt sich das Ergebnis mit dem ersten Beispiel, der Schnittpunktbestimmung zweier Geraden.

Der Fall $|f'(x^*)| \ll 1$ entspricht gerade dem schleifenden Schnitt, siehe Kapitel 2.2.2.

Bemerkung 2.2.6 Man beachte, daß der Auswertungsfehler von f von der gewählten Darstellung von f abhängt. So ist trivialerweise $\varepsilon = 0$ bei der Auswertung eines Polynoms

$$p(x) = (x - x_1) \cdot \ldots \cdot (x - x_n) = 0$$

in Wurzeldarstellung. Das gleiche Polynom in Koeffizientendarstellung führt im allgemeinen zu $\varepsilon \neq 0$.

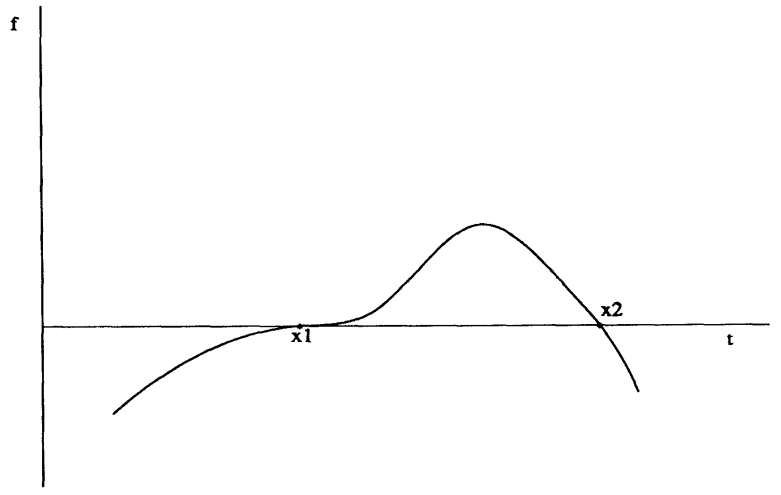


Abbildung 2.4: schlecht konditionierte Nullstelle bei x_1 , gut konditionierte bei x_2

2.2.5 Berechnung der Eigenwerte von symmetrischen Matrizen

Die obigen Überlegungen zur Auswertung und Nullstellenbestimmung von Polynomen in Koeffizientenschreibweise scheinen zunchst darauf hinzudeuten, daß die Bestimmung der Eigenwerte einer Matrix (als Wurzeln des charakteristischen Polynoms) generell schlecht konditioniert ist. Wie wir im folgenden sehen werden, ist dies für die wichtige Klasse der symmetrischen Matrizen jedoch nicht der Fall. Lineare Eigenwertprobleme der Form

$$Ax = \lambda x, \quad A \text{ reell und symmetrisch}$$

spielen in den Anwendungen eine große Rolle. Ersetzt man in dem Eigenwertproblem für Differentialgleichungen

$$-u'' = \lambda u, \quad u(0) = u(1) = 0,$$

welches zum Beispiel die Eigenschwingungen einer Saite beschreibt, die Ableitungen näherungsweise durch Differenzenquotienten (Diskretisierung) so ergeben sich lineare Eigenwertprobleme

$$A_h U_h = \lambda U_h$$

mit symmetrischen Matrizen A_h . Aus der linearen Algebra sind folgende Eigenschaften symmetrischer Matrizen bekannt:

Lemma 2.2.7 *Sei* $A \in Mat_n(\mathbf{R})$ *eine reelle symmetrische Matrix. Dann gilt:*

a) Es existieren genau n Eigenwerte $\lambda_1, \ldots, \lambda_n$ *(mit Vielfachheit gezählt),*

b) alle Eigenwerte sind reell, $\lambda_i \in \mathbf{R}$,

c) es gibt eine Orthonormalbasis $\eta_1, \ldots, \eta_n \in \mathbf{R}^n$ *von Eigenvektoren.*

Mit diesen Bezeichnungen gilt also

$$\begin{aligned} A\eta_i &= \lambda_i \eta_i \text{ für } i = 1, \ldots, n \\ \eta_i^T \eta_j &= \delta_{ij} \text{ für } i, j = 1, \ldots, n \end{aligned}$$

Die aus den Eigenvektoren als Spalten bestehende Matrix

$$Q := (\eta_1, \ldots, \eta_n) \in O(n)$$

ist orthogonal, d.h. $QQ^T = Q^T Q = I$, und mit

$$\Lambda := \text{diag}\,(\lambda_1, \ldots, \lambda_n)$$

gilt

$$\begin{aligned} A &= Q\Lambda Q^T, \quad \Lambda = Q^T A Q \\ \lambda_i &= \eta_i^T A \eta_i \text{ für } i = 1, \ldots, n \end{aligned}$$

Eine Störung der Eingabegröße A läßt sich beschreiben durch

$$A \to \tilde{A} := A + \Delta A, \quad \Delta A \in \text{Mat}_n(\mathbf{R}),$$

wobei aber nur symmetrische Störungen $\Delta A = (\Delta A)^T$ sinnvoll sind. (Speichern wir die symmetrische Matrix A in einem Rechner, so legen wir natürlich nur eine der beiden identischen Hälften ab, so daß sich z.B. Rundungsfehler auf beide Hälften gleich auswirken.)

Wir betrachten daher als Eingabemenge

$$\begin{aligned} E &:= \{\tilde{A} \mid \tilde{A} = A + \Delta A, \ \|\Delta A\|_2 \le \varepsilon, \ (\Delta A)^T = \Delta A\} \\ [E]_{\text{abs}} &= \sup_{\tilde{A} \in E} \|A - \tilde{A}\|_2 = \varepsilon \end{aligned}$$

Daß wir gerade die Spektralnorm zur Beschreibung der Größe der Störung ΔA gewählt haben, wird sich im nachhinein als günstig erweisen.

Die Matrix $\tilde{A}$ habe die Eigenwerte $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n$ und orthonormalen Eigenvektoren $\tilde{\eta}_1, \ldots, \tilde{\eta}_n$. Dann gilt wegen

$$\begin{aligned} \tilde{A}\tilde{\eta}_i &= (A + \Delta A)\tilde{\eta}_i = \tilde{\lambda}_i \tilde{\eta}_i \\ \Delta A \tilde{\eta}_i &= (\tilde{\lambda}_i I - A)\tilde{\eta}_i \end{aligned}$$

daß

$$(Q^T(\tilde{\lambda}_i I - A)Q)Q^T\tilde{\eta}_i = (\underbrace{\tilde{\lambda}_i I - \Lambda}_{=:\hat{\Lambda}})\underbrace{Q^T\tilde{\eta}_i}_{=:\hat{\eta}_i} = (\underbrace{Q^T \Delta A\, Q}_{\Delta\hat{A}})\underbrace{Q^T\tilde{\eta}_i}_{=\hat{\eta}_i}$$

und daher

$$\hat{\Lambda}\hat{\eta}_i = \Delta\hat{A}\hat{\eta}_i$$

wobei

$$\hat{\Lambda} = \tilde{\lambda}_i I - \Lambda = \text{diag}\,(\tilde{\lambda}_i - \lambda_1, \ldots, \tilde{\lambda}_i - \lambda_n)$$

Falls $\hat{\Lambda}$ singulär ist, so gilt $\tilde{\lambda}_i = \lambda_j$ für ein $j \in \{1, \ldots, n\}$, d.h. λ_j ist unverändert. Falls $\hat{\Lambda}$ regulär ist, so gilt

$$\|\hat{\eta}_i\|_2 = \|\hat{\Lambda}^{-1}\Delta\hat{A}\hat{\eta}_i\|_2 \leq \|\hat{\Lambda}^{-1}\|_2\|\Delta\hat{A}\|_2\|\hat{\eta}_i\|_2$$

Da die Euklidische Norm invariant unter orthogonalen Transformationen ist, gilt

$$\|\hat{\eta}_i\|_2 = \|Q^T\tilde{\eta}_i\|_2 = \|\tilde{\eta}_i\|_2 = 1$$

$$\|\Delta\hat{A}\|_2 = \|Q^T\Delta AQ\|_2 = \|\Delta A\|_2$$

und für $\|\hat{\Lambda}^{-1}\|$ erhalten wir

$$\|\hat{\Lambda}^{-1}\|_2 = \max_{j=1,\ldots,n} \frac{1}{|\tilde{\lambda}_i - \lambda_j|} = \frac{1}{\min\limits_j |\tilde{\lambda}_i - \lambda_j|}$$

Aus $(\cdot)$ folgt daher

$$\min_j |\tilde{\lambda}_i - \lambda_j| \leq \|\Delta A\|_2 \text{ für } i = 1, \ldots, n$$

Somit ist

$$\kappa_{\text{abs}} = 1$$

und das Eigenwertproblem für symmetrische Matrizen in Hinblick auf absolute Fehler der Eigenwerte gut konditioniert. (Die Kondition der Berechnung der Eigenvektoren wird hier nicht untersucht.)

Bemerkung 2.2.8 Die Behauptung wird falsch, wenn A nicht symmetrisch ist. Für die Matrizen

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad \tilde{A} = \begin{pmatrix} 0 & 1 \\ \varepsilon & 0 \end{pmatrix},$$

gilt z. B. $\|\Delta A\|_2 = \varepsilon\|A\|_2 = \varepsilon$ mit den Eigenwerten

$$\begin{aligned} &\text{Eigenwerte von} \quad A: \quad \lambda_1 = 0, \quad \lambda_2 = 0 \\ &\text{Eigenwerte von} \quad \tilde{A}: \quad \tilde{A}_1 = \sqrt{\varepsilon}, \quad \tilde{\lambda}_2 = -\sqrt{\varepsilon} \end{aligned}$$

Daraus ergibt sich für die Kondition des Eigenwertproblems

$$\frac{|\lambda_1 - \tilde{\lambda}_1|}{\|\Delta A\|_2} = \frac{\sqrt{\varepsilon}}{\varepsilon} = \frac{1}{\sqrt{\varepsilon}} \to \infty \quad \text{für} \quad \varepsilon \to 0$$

2.3 Stabilität eines Algorithmus

Wir wenden uns nun der zweiten Gruppe von Fehlern zu, den Fehlern im Algorithmus, wobei wir uns in diesem Kapitel auf Rundungsfehler beschränken. Bei der Ausführung der Berechnung wird aufgrund von Rundungsfehlern eine Abbildung $\tilde{f}$ anstelle von f realisiert, die die Eingabe e auf $\tilde{r} = \tilde{f}(e)$ abbildet.

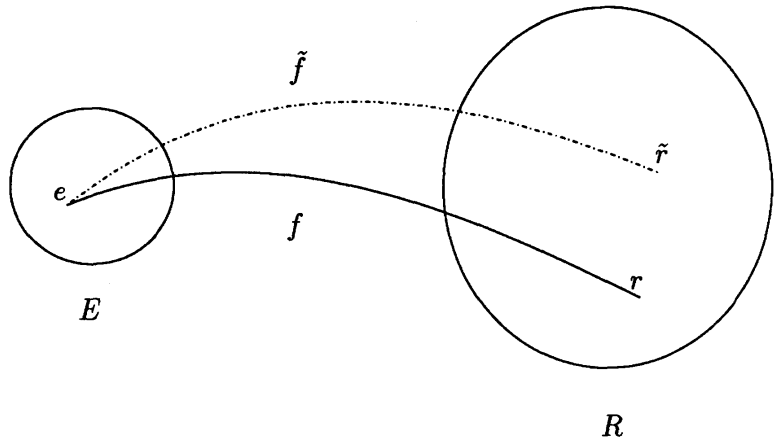


Abbildung 2.5: "exaktes" Ergebnis $r = f(e)$, berechnetes Ergebnis $\tilde{r} = \tilde{f}(e)$

Die Frage nach der *Stabilität* des Algorithmus lautet nun:

"Ist das Resultat $\tilde{r}$ anstelle von r akzeptabel ?"

Wir haben oben gesehen, daß sich die Fehler bei der Ausführung einer elementaren Gleitkommaoperation $\circ \in \{+, -, *, /\}$ abschätzen läßt durch

$$\mathrm{fl}(\bar{a} \circ \bar{b}) = (\bar{a} \circ \bar{b})(1 + \varepsilon) \text{ mit } |\varepsilon| \leq \text{ eps} \tag{2.3.1}$$

Dabei wissen wir nicht, wie ε für spezielle Werte für $\bar{a}$ und $\bar{b}$ aussieht. (Man könnte natürlich ε prinzipiell für alle $\bar{a}, \bar{b}$ auf dem speziellen Rechner bestimmen, was jedoch den Rahmen jeder vernünftigen Fehlertheorie sprengen würde.) Daher haben wir es nicht mit einer einzigen Abbildung $\tilde{f}$ zu tun, sondern mit einer ganzen Klasse $F = \{\tilde{f}\}$, die alle durch eine Abschätzung der Art $(\cdot)$ charakterisierten Abbildungen umfaßt. Für eine Elementaroperation erhalten wir:

$$F = \{\tilde{f} \mid \forall a, b \ \exists \varepsilon \text{ mit } |\varepsilon| \leq \text{ eps und } \tilde{f}(a, b) = f(a, b)(1 + \varepsilon)\}$$

Zu dieser Klasse von Funktionen gehört insbesondere auch die "exakte" Abbildung f, also $f \in F$. Um den Begriff der Stabilität zu präzisieren, gibt es im wesentlichen zwei Ansätze, die *Vorwärtsanalyse* und die *Rückwärtsanalyse.*

2.3.1 Vorwärtsanalyse (forward analysis)

Wir gehen von den in Kapitel 2.2 eingeführten (problemspezifisch definierten) Eingabe- bzw. Resultatmengen E und R aus.

Eine fehlerbehaftete Realisierung $\tilde{f} \in F$ von f bildet E auf eine Menge $\tilde{f}(E)$ von Resultaten ab. Die Vereinigung

$$\tilde{R} := \bigcup_{\tilde{f} \in F} \tilde{f}(E)$$

aller dieser Mengen $\tilde{f}(E)$ charakterisiert die Wirkung des Algorithmus auf die Eingabemenge E. Insbesondere gilt wegen $f \in F$, daß

$$f(E) = R \subset \tilde{R}\,.$$

Geben wir wie in Kapitel 2.2 ein Fehlermaß $[\cdot]$ zur Charakterisierung der "Größe" einer Menge von Resultaten vor (z.B. $[\cdot]_{\text{abs}}$ oder $[\cdot]_{\text{rel}}$), so können wir definieren:

Definition 2.3.1 *Ein Algorithmus heißt* in Vorwärtsanalyse stabil, *falls*

$$[\tilde{R}] \approx [R],$$

und in Vorwärtsanalyse instabil, *falls*

$$[\tilde{R}] \gg [R].$$

Wir werden unten sehen, daß bei der Ausführung von n Elementaroperationen $(+, -, \cdot, /)$ ein Fehler von der Größenordnung $[\tilde{R}] = O(n)[R]$ unvermeidbar ist. Eine vernünftige Abschwächung der Stabilitätsbedingung ist daher:

Definition 2.3.2 *Ein Algorithmus heißt* in abgeschwächter Form stabil *(in Vorwärtsanalyse), falls*

$$[\tilde{R}] = O(n)[R],$$

wobei n die Anzahl der hintereinander ausgeführten Elementaroperationen ist.

2.3.2 Rückwärtsanalyse (backward analysis)

Bei dieser von Wilkinson eingeführten Fehleranalyse faßt man die fehlerbehafteten Resultate $\tilde{r} = \tilde{f}(e)$ als "exakte" Ergebnisse zu gestörten Eingabegrößen $\tilde{e}$ auf, d.h.

$$f(\tilde{e}) = \tilde{r} = \tilde{f}(e)$$

Wir definieren $\tilde{E}$ durch

$$\tilde{E} := \{\tilde{e} \mid \; f(\tilde{e}) = \tilde{f}(e), \quad \tilde{f} \in F, \quad e \in E\}$$

Wegen $f \in F$ gilt $E \subset \tilde{E}$, so daß wir die Stabilität in Rückwärtsanalyse durch den Vergleich von $[\tilde{E}]$ und $[E]$ definieren können, wobei $[\cdot]$ wieder die "Größe" einer Menge von Eingabedaten charakterisiert.

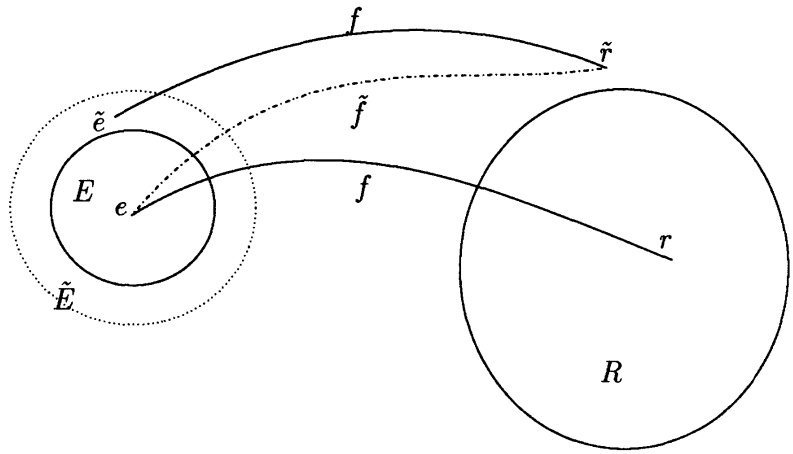


Abbildung 2.6: Eingabe- und Resultatmengen bei der Rückwärtsanalyse

Definition 2.3.3 *Ein Algorithmus heißt in* in Rückwärtsanalyse

a) stabil, *falls*

$$[\tilde{E}] \approx [E],$$

b) in abgeschwächter Form stabil, *falls*

$$[\tilde{E}] = O(n)[E],$$

c) *und* instabil, *falls*

$$[\tilde{E}] \gg [E]$$

Die Rückwärtsanalyse ist natürlich nur dann sinnvoll, wenn sich tatsächlich alle fehlerhaften Resultate $\tilde{r}$ als Bilder $f(\tilde{e})$ von gestörten Eingabedaten schreiben lassen. Ist dies jedoch möglich, so erweist sich die Rückwärtsanalyse als einfacher als die Vorwärtsanalyse, da sie keine Konditionsanalyse verlangt. Daraus folgt auch, daß nur auf der Eingabeseite ein vernünftiges Fehlermaß eingeführt werden muß, welches sich meistens aus der Art der auftretenden Eingabefehler leicht ableiten läßt. Wir führen im folgenden beide Techniken an etlichen Beispielen vor.

2.3.3 Ausführung der Subtraktion

Wir wollen die Differenz $r = f(a,b) = a - b, \quad a > b > 0$ berechnen. Wie wir bei der Konditionsanalyse gesehen haben, ist

$$\begin{aligned} E &= \{(a,b) \mid a = \frac{\bar{a}}{1+\varepsilon_a}, \quad b = \frac{\bar{b}}{1+\varepsilon_b} \text{ mit } |\varepsilon_a|, |\varepsilon_b| \leq \text{ eps}\} \\ R &= \{r = a - b | (a,b) \in E\} \\ [E]_{\text{rel}} &= \text{eps} \\ [R]_{\text{rel}} &= \frac{|a| + |b|}{|a-b|} \text{ eps} \end{aligned}$$

Anstelle von $r = a - b$ produziert der Rechner für $(a,b) \in E$ die Lösung

$$\tilde{r} = \text{fl}(a - b) = (a - b)(1 + \alpha)$$

mit einem relativen Subtraktionsfehler α mit $|\alpha| \leq$ eps. Für die Fehleranalyse wird unser "Subtraktionsalgorithmus" beschrieben durch die Klasse $F = \{\tilde{f}\}$ von Abbildungen

$$F = \{\tilde{f} \mid \tilde{f} : (a,b) \mapsto (a-b)(1-\alpha_{a,b}) \text{ mit } |\alpha_{a,b}| \leq \text{ eps}\}$$

und führt zu der Menge der gestörten Resultate

$$\begin{aligned} \tilde{R} &= \bigcup_{\tilde{f} \in F} \tilde{f}(E) \\ &= \{\tilde{r} \mid \tilde{r} = (a-b)(1+\alpha) \text{ mit } (a,b) \in E, \ |\alpha| \leq \text{ eps}\} \end{aligned}$$

Nun ist für $(a,b) \in E$ mit $a = \bar{a}/(1+\varepsilon_a)$ und $b = \bar{b}/(1+\varepsilon_b)$

$$\begin{aligned} \tilde{r} &= (a-b)(1+\alpha) \\ &= \bar{a}\left(\frac{1+\alpha}{1+\varepsilon_a}\right) - \bar{b}\left(\frac{1+\alpha}{1+\varepsilon_b}\right) \\ &= \bar{a}(1+\tilde{\varepsilon}_a) - \bar{b}(1+\tilde{\varepsilon}_b) \\ &= (\bar{a}-\bar{b})\left(1 + \frac{\bar{a}}{\bar{a}-\bar{b}}\tilde{\varepsilon}_a - \frac{\bar{b}}{\bar{a}-\bar{b}}\tilde{\varepsilon}_b\right), \end{aligned}$$

wobei

$$\tilde{\varepsilon}_{a,b} := \frac{1+\alpha}{1+\varepsilon_{a,b}} - 1 = \frac{\alpha - \varepsilon_{a,b}}{1+\varepsilon_{a,b}} \doteq \alpha - \varepsilon_{a,b} \text{ mit } |\tilde{\varepsilon}_{a,b}| \leq 2 \text{ eps}$$

a) **Vorwärtsanalyse**

Wir erhalten so bei Vernachlässigung der $O(\text{eps}^2)$-Terme (linearisierte Fehlertheorie)

$$[\tilde{R}]_{\text{rel}} = \sup_{\tilde{r} \in \tilde{R}} \frac{|\tilde{r} - \bar{r}|}{|\tilde{r}|} = \frac{|\bar{a}| + |\bar{b}|}{|\bar{a} - \bar{b}|} 2 \text{ eps}$$

im Vergleich zu

$$[R]_{\text{rel}} = \sup_{r \in R} |\frac{r - \bar{r}}{r}| = \frac{|\bar{a}| + |\bar{b}|}{|\bar{a} - \bar{b}|} \text{ eps}$$

also

$$[\tilde{R}]_{\text{rel}} = 2[R]_{\text{rel}}.$$

Die Subtraktion ist also zumindest im abgeschwächten Sinn vorwärtsstabil.

b) **Rückwärtsanalyse:**

Nach (.) und (.) ist die Rückwärtsanalyse anwendbar und es gilt

$$\tilde{E} = \{(\tilde{a}, \tilde{b}) \mid \tilde{a} = \bar{a}(1+\tilde{\varepsilon}_a),\ \tilde{b} = \bar{a}(1+\tilde{\varepsilon}_b) \text{ mit } |\tilde{\varepsilon}_a|, |\tilde{\varepsilon}_b| \leq 2 \text{ eps}\}$$

also

$$[\tilde{E}]_{\text{rel}} = 2 \text{ eps} = 2[E]_{\text{rel}}.$$

Dies liefert uns die Stabilität der Subtraktion, ohne daß wir auf die Resultate der Konditionsanalyse zurückgreifen müssen.

Bemerkung 2.3.4 Im schlechtkonditionierten Fall der Auslöschung ist die Subtraktion im Rechner sogar fehlerfrei, also $\alpha = 0$, da das Ergebnis nicht gerundet werden muß !

2.3.4 Ausführung des Skalarproduktes

Bei der Konditionsanalyse für das Skalarprodukt $< a, b >$ zweier Vektoren $a, b \in \mathbf{R}^n$ haben wir hergeleitet, daß

$$\kappa_{\text{rel}} = \frac{[R]_{\text{rel}}}{[E]_{\text{rel}}} = \frac{|\bar{a}|^T|\bar{b}|}{|\bar{a}^T\bar{b}|}$$

Zur Berechnung des Skalarproduktes analysieren wir den denkbar einfachsten Algorithmus

$$\begin{aligned} s_0 &:= 0 \\ s_i &:= s_{i-1} + a_i * b_i \text{ für } i = 1, \ldots, n \end{aligned}$$

Im Rechner werden statt der "exakten" Werte $s_0, \ldots, s_n$ die fehlerbehafteten Größen $\tilde{s}_0, \ldots, \tilde{s}_n$ berechnet

$$\begin{aligned} \tilde{s}_0 &= 0 \\ \tilde{s}_1 &= \text{fl}(a_1 b_1) = a_1 b_1 (1+\mu_1) \\ \tilde{s}_2 &= \text{fl}(\tilde{s}_1 + \text{fl}(a_2 b_2)) = (\tilde{s}_1 + a_2 b_2 (1+\mu_2))(1+\alpha_2) \\ &\vdots \\ \tilde{s}_i &= \text{fl}(\tilde{s}_{i-1} + \text{fl}(a_i b_i)) = (\tilde{s}_{i-1} + a_i b_i (1+\mu_i))(1+\alpha_i), \end{aligned}$$

wobei die μ_i, $|\mu_i| \leq$ eps, die relativen Multiplikationsfehler und die α_i, $|\alpha_i| \leq$ eps, die relativen Additionsfehler beschreiben. Durch Induktion über i folgt

$$\tilde{s}_n = \sum_{i=1}^{n} a_i b_i (1+\gamma_i) \text{ mit } \gamma_i := (1+\mu_i)(1+\alpha_i)\cdots(1+\alpha_n) - 1$$

also für $(a,b) \in E$, $a_i = \bar{a}_i/(1+\varepsilon_i)$, $b_i = \bar{b}_i/(1+\delta_i)$:

$$\tilde{s}_n = \sum_{i=1}^{n} \bar{a}_i \bar{b}_i (1+\kappa_i)$$

mit

$$\begin{aligned} \kappa_i &:= (1+\varepsilon_i)^{-1}(1+\delta_i)^{-1}(1+\gamma_i) - 1 \\ &= (1+\varepsilon_i)^{-1}(1+\delta_i)^{-1}(1+\mu_i)(1+\alpha_i)\cdots(1+\alpha_n) - 1 \end{aligned}$$

Zur Abschätzung dieses Ausdrucks verwenden wir folgendes Lemma:

Lemma 2.3.5 *Falls* $|\delta_i| \leq \text{eps}$ *und* $p_i \in \{\pm 1\}$ *für* $i = 1,\ldots,n$ *und* $n \cdot \text{eps} < 1$, *so gilt*

$$\prod_{i=1}^{n}(1+\delta_i)^{P_i} = 1+\theta_n \quad \textit{mit} \quad |\theta_n| \leq \gamma_n := \frac{n \cdot \text{eps}}{1 - n \cdot \text{eps}}$$

Beweis: s. Aufgabe ? □

Damit gilt für κ_i

$$|\kappa_i| \leq \frac{(n+4-i)\,\text{eps}}{1-(n+4-i)\,\text{eps}} \doteq (n+4-i)\,\text{eps}$$

a) **Vorwärtsanalyse**

Zur Berechnung von $[\tilde{R}]_{\text{rel}}$ beachten wir, daß

$$\begin{aligned} |\tilde{s}_n - \bar{s}_n| &= |\textstyle\sum_{i=1}^{n} \bar{a}_i\bar{b}_i(1+\kappa_i) - \sum_{i=1}^{n} \bar{a}_i\bar{b}_i| \\ &= |\textstyle\sum_{i=1}^{n} \bar{a}_i\bar{b}_i\kappa_i| \\ &\leq \max_j \kappa_j \textstyle\sum_{i=1}^{n} |\bar{a}_i\bar{b}_i| \\ &\leq (n+3)\,\text{eps}|\bar{a}|^T|\bar{b}| \end{aligned}$$

und damit

$$[\tilde{R}]_{\text{rel}} = \sup_{\tilde{r}\in\tilde{R}} \frac{|\tilde{r}-\bar{r}|}{|\bar{r}|} \doteq (n+3)\,\text{eps}\frac{|\bar{a}|^T|\bar{b}|}{|\bar{a}^T\bar{b}|}$$

Der Vergleich mit der relativen Kondition zeigt, daß unser Algorithmus in abgeschwächter Form vorwärtsstabil ist.

b) **Rückwärtsanalyse**

Für die Rückwärtsanalyse müssen wir die Menge $\tilde{E}$ der Eingabegrößen $(\tilde{a},\tilde{b})$ betrachten, die bei exakter Rechnung auf ein Resultat $\tilde{s}_n \in \tilde{R}$ führen. Nach (.) ist

$$\tilde{E} = \{(\tilde{a},\tilde{b}) \mid \tilde{a}_i\tilde{b}_i = \bar{a}_i\bar{b}_i(1+\kappa_i) \text{ mit } |\kappa_i| \leq (n+4-i)\,\text{eps}\}$$

und daher

$$\begin{aligned}[\tilde{E}]_{\text{rel}} &= \sup_{(\tilde{a},\tilde{b})\in\tilde{E}} \max_i \frac{|\tilde{a}_i\tilde{b}_i - \bar{a}_i\bar{b}_i|}{|\tilde{a}_i\tilde{b}_i|} \\ &\doteq \max_i (n+4-i)\ \text{eps} \\ &= (n+3)\ \text{eps}\end{aligned}$$

Der Vergleich mit $[E]_{\text{rel}} \doteq 2$ eps zeigt, daß der Algorithmus in abgeschwächter Form rückwärtsstabil ist.

2.4 Differentielle Fehleranalyse

Wir wollen in diesem Abschnitt eine weitere Methode der Fehlerrechnung, die sogenannte *differentielle Fehleranalyse* kennenlernen. Dabei zerlegen wir die Abbildung $f : e \mapsto r$ in eine Kette elementarer Operationen $f_0, \ldots, f_n$,

$$\begin{aligned}e &= \alpha_0 \overset{f_0}{\mapsto} \alpha_1 \mapsto \cdots \overset{f_n}{\mapsto} \alpha_{n+1} = r \\ f &= f_n \circ f_{n-1} \circ \cdots \circ f_0\end{aligned}$$

die den Algorithmus näher beschreiben, wobei $\alpha_i \in \mathbf{R}^{n_i}$ vom Algorithmus benutzte Zwischengrößen sind. Für unsere Überlegungen nehmen wir an, daß die Abbildungen $f_0, \ldots, f_n$ stetig differenzierbar sind. Im Rechner werden diese Abbildungen durch gestörte Abbildung $\tilde{f}_i$ realisiert, deren absolute Fehler bei der Berechnung von $f_i(\alpha_i)$ wir mit $\Delta\alpha_{i+1}$ bezeichnen, d.h.

$$\tilde{f}_i(\alpha_i) - f_i(\alpha_i) = \Delta\alpha_{i+1} \quad \text{für} \quad i = 0, \ldots, n$$

Um die Fortpflanzung des Eingabefehlers

$$\tilde{e} - e = \Delta e = \Delta\alpha_0$$

zu beschreiben, benötigen wir die *Restabbildungen*

$$\begin{aligned}\psi_{n-k} &:= f_n \circ \cdots \circ f_k \\ \varphi_k &:= f_k \circ \cdots \circ f_0\end{aligned} \qquad \text{für} \quad k = 0, \ldots, n$$

Gehen wir nun davon aus, daß sich die Fehler (im Sinne einer linearisierten Fehlertheorie) linear überlagern (eine Art Superpositionsprinzip der linearen Fehlertheorie), so gilt für den Gesamtfehler Δr, daß

$$\Delta r \doteq \sum_{k=0}^{n} D\psi_{n-k}(\alpha_k)\Delta\alpha_k + \Delta\alpha_{n+1} \tag{2.4.1}$$

wobei jeweils der erste Summand

$$D\psi_n(\alpha_0)\Delta\alpha_0 = Df(e)\Delta e$$

nur vom Problem f abhängt und daher der Kondition zuzuschreiben ist, wohingegen die übrigen Summanden die vom Algorithmus eingeschleppten Fehler beschreiben. Wir beschränken uns im folgenden auf den skalaren Fall, $\alpha_0, \ldots, \alpha_{m+1} \in \mathbf{R}$, und bezeichnen mit ε_{i+1} den relativen Fehler bei der Berechnung von $f_i(\alpha_i)$, d. h.

$$\tilde{f}_i(\alpha_i) - f_i(\alpha_i) = \Delta\alpha_{i+1} = \varepsilon_{i+1} f_i(\alpha_i) \quad \text{für} \quad i = 0, \ldots, n$$

Analog zu (2.4.1) gilt dann für den Gesamtfehler Δr, daß

$$\Delta r \doteq \sum_{k=0}^{n} \psi'_{n-k}(\alpha_k)\varepsilon_k\alpha_k + \varepsilon_{n+1}\alpha_{n+1}, \tag{2.4.2}$$

mit dem Konditionsanteil

$$\psi'_n(\alpha_0)\Delta\alpha_0 = f'(e)e\varepsilon_0.$$

Falls

$$\psi'_n(\alpha_0) = f'(e)e \neq 0,$$

erhalten wir wegen

$$\begin{aligned} \psi'_n(\alpha_0) &= (f_n \circ \cdots \circ f_0)'(\alpha_0) \\ &= (f_n \circ \cdots \circ f_k)'(\alpha_k) \cdot (f_{k-1} \circ \cdots \circ f_0)'(\alpha_0) \\ &= \psi'_{n-k}(\alpha_k) \cdot \varphi'_{k-1}(\alpha_0) \end{aligned}$$

für den Gesamtfehler Δr die Gleichung

$$\begin{aligned} \Delta r &\doteq \psi'_n(\alpha_0)\left(\varepsilon_0 + \sum_{k=1}^{n}(\alpha_0)\frac{\psi'_{n-k}(\alpha_k)\alpha_k}{\psi'_n(\alpha_0)\alpha_0}\varepsilon_k + \frac{\alpha_{n+1}}{\psi'_n(\alpha_0)\alpha_0}\varepsilon_{n+1}\right) \\ &= f'(\alpha_0)\alpha_0\left(\varepsilon_0 + \sum_{k=1}^{n+1}\frac{\alpha_k}{\varphi'_{k-1}(\alpha_0)\alpha_0}\varepsilon_k\right) \\ &= f'(e)e\left(\varepsilon_0 + \sum_{k=1}^{n+1}\frac{\varphi_{k-1}(e)}{\varphi'_{k-1}(e)\cdot e}\varepsilon_k\right) \end{aligned} \tag{2.4.3}$$

Die *Verstärkungsfaktoren*

$$\mu_k = \mu_k(e) := \left|\frac{\varphi_{k-1}(e)}{\varphi'_{k-1}(e)e}\right| \tag{2.4.4}$$

geben Auskunft über die Propagation des Fehlers des k-ten Zwischenschrittes.

Wie man sich denken kann, ist eine differentielle Fehleranalyse aller Zwischenschritte eines Algorithmus sehr aufwendig. Daher greift man meistens nur wenige kritische Zwischenschritte für die Analyse heraus. Für einen einzigen Zwischenschritt $n = 1$, $f = h \circ g$

$$x \overset{g}{\mapsto} y = g(x) \overset{h}{\mapsto} z = h(y)$$

lichten sich die Formeln zu

$$\Delta r \doteq f'(x)x(\varepsilon_x + \underbrace{\frac{g(x)}{g'(x)x}}_{=\mu(x)}\varepsilon_g) + z\varepsilon_h$$

wobei wir mit

$$\Delta x = x\varepsilon_x, \quad \Delta y = y\varepsilon_g, \quad \Delta z = z\varepsilon_h$$

den Eingabefehler und die Fehler bei der Ausführung von g bzw. h bezeichnen.

Beispiel 2.4.1 Rekursive Berechnung von $c_m := \cos(mx)$

Aufgrund des Additionstheorems

$$\cos((k+1)x) = 2\cos x \cos(kx) - \cos((k-1)x)$$

läßt sich $c_m = \cos(mx)$ mit Hilfe der 3-Term-Rekursion

$$\begin{aligned} c_0 &:= 1, \\ c_1 &:= \cos x \\ c_{k+1} &= 2c_k \cos x - c_{k-1} \quad \text{für } k = 1, 2, \ldots \end{aligned}$$

berechnen. Der Fehler ε_g bei der Auswertung von

$$g(x) := 2\cos x$$

geht in jeden Zwischenschritt ein. Zur Beurteilung der Stabilitätseigenschaften der 3-Term-Rekursion sehen wir uns daher den Verstärkungsfaktor

$$\mu(x) = \left|\frac{g(x)}{g'(x)x}\right| = \left|\frac{2-\cos x}{2x\sin x}\right| = \left|\frac{1}{x\tan x}\right|$$

an. Es gilt

$$\begin{aligned} x \to 0 \quad &\Rightarrow \quad \mu(x) \to \frac{1}{x^2} \to \infty \\ x \to \pi \quad &\Rightarrow \quad \mu(x) \to \frac{1}{\pi(x-\pi)} \to \infty \end{aligned}$$

In beiden Grenzfällen ist die 3-Term-Rekursion instabil (wobei der erste Fall $x \to 0$ der kritischere ist).

Es gibt jedoch eine stabile Rekursion zur Berechnung von $\cos(mx)$, die von C. REINSCH entwickelt wurde. Dazu führt man die Differenzen

$$\Delta c_k := c_{k+1} - c_k$$

ein und erhält durch Einsetzen von (.)

$$\begin{aligned} \Delta c_k &= 2c_k \cos x - c_{k-1} - c_k \\ &= 2(\cos x - 1)c_k + \Delta c_{k-1} \\ &= -4c_k \sin^2(x/2) + \Delta c_{k-1} \end{aligned}$$

und damit die Rekursion

$$\begin{aligned} c_0 &:= 1, \\ \Delta c_0 &:= -2\sin^2(x/2) \\ c_k &:= c_{k-1} + \Delta c_{k-1} \\ \Delta c_k &:= -4c_k \sin^2(x/2) + \Delta c_{k-1} \end{aligned}$$

Hier geht in jeden Zwischenschritt

$$g(x) := -4\sin^2(x/2)$$

und für den Verstärkungsfaktor ergibt sich

$$\mu(x) = \left|\frac{\sin^2(x/2)}{x \cdot 2\sin(x/2)\cos(x/2)/2}\right| = \left|\frac{\tan(x/2)}{x}\right|$$

Für den ersten kritischen Grenzfall gilt

$$x \to 0 \Rightarrow \mu(x) \to \frac{1}{2}$$

d.h. der Zwischenschritt ist stabil.

Bemerkung 2.4.2 1. Für $\cos x < 0$ wählt man eine andere Form der Stabilisierung ausgehend von $\Delta c_k := c_{k+1} + c_k$.

2. Die wichtigste Anwendung ist die Berechnung von $\cos(mx)$ für die Fourier-Synthese. Hier ist häufig
$$x = \frac{2\pi}{N} \text{ mit } N = 2^n \text{ und } m = 1, \ldots, N$$

2. Die Stabilisierungstechnik ist nur erfolgreich, weil die trigonometrische Rekursion gutkonditioniert ist.

2.5 Fehleranalyse für lineare Gleichungssysteme

In diesem Abschnitt wollen wir die Fehleranalyse auf lineare Gleichungssysteme der Form $Ax = b$ und deren Lösung mit Hilfe der Gauß-Elimination anwenden. Dazu genügt es, wie wir oben dargelegt haben, die Kondition des Problems und die Stabilität des Algorithmus in Rückwärtsanalyse zu untersuchen. Bei den Eingabegrößen A, b haben wir sowohl Fehler der Matrix A als auch der rechten Seite b zu berücksichtigen:

$$\begin{array}{lcll} \text{Eingabe} & : & (a,b) & \to & (\bar{A},\bar{b}) \\ \text{Resultat} & : & x & \to & \tilde{x} \end{array}$$

Zum Vergleich von Fehlern von Vektoren und Matrizen benötigen wir zum ersten Mal Normen auf Vektorräumen. Die Eigenschaften einer Norm $\|\cdot\|$ auf einem Vektorraum X über dem Körper $\mathbf{K} = \mathbf{R}, \mathbf{C}$

a) $\|x\| \geq 0$ für alle $x \in X$

b) $\|x\| = 0 \iff x = 0$

c) $\|\alpha x\| = |\alpha|\,\|x\|$ für alle $x \in X$ und $\alpha \in \mathbf{K}$

d) $\|x + y\| \leq \|x\| + \|y\|$ für alle $x, y \in X$

seien als bekannt vorausgesetzt. Wir beschränken uns hier im wesentlichen auf die sogenannten *p-Normen* auf $\mathbf{R}^n$

$$\begin{aligned} \|x\|_p &:= (\textstyle\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}} \text{ für } 1 \leq p < \infty \\ \|x\|_\infty &:= \max_i |x_i|, \text{ (Maximumsnorm)} \end{aligned}$$

Die 1-Norm wird auch *Betragssummennorm* und die 2-Norm *Euklidische Norm* genannt, da sie wegen

$$\|x\|_2 = \sqrt{< x, x >}$$

von dem Euklidischen Skalarprodukt induziert wird. Für Matrizen verwenden wir die zugeordneten Operatornormen

$$\|A\|_p := \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$$

Für diese gilt

a) $\|I\|_p = 1$

b) $\|AB\|_p \leq \|A\|_p \|B\|_p$ (Submultiplikativität)

c) $\|Ax\|_p \leq \|A\|_p \|x\|_p$ (Verträglichkeit mit der Vektornorm)

2.5.1 Kondition

Sei x die exakte Lösung von $Ax = b$ und $\tilde{x} = x + \Delta x$ die exakte Lösung des gestörten Problems

$$\tilde{A}\tilde{x} = \tilde{b}, \quad \tilde{A} = A + \Delta A, \quad \tilde{b} = b + \Delta b$$

Für die relativen Fehler führen wir die Bezeichnungen

$$\varepsilon(A) := \frac{\|\Delta A\|}{\|A\|}, \quad \varepsilon(b) := \frac{\|\Delta b\|}{\|b\|}, \quad \varepsilon(x) = \frac{\|\Delta x\|}{\|x\|}$$

ein, wobei $A, b, x \neq 0$ vorausgesetzt sei. Weiter definieren wir

$$\kappa(A) := \|A\| \cdot \|A^{-1}\|$$

falls A nicht singulär ist und zeigen:

Satz 2.5.1 *Sei $A \in Mat_n(\mathbf{R})$ nicht singulär und $1 - \kappa(A)\varepsilon(A) > 0$. Dann gilt*

$$\varepsilon(x) \leq \frac{\kappa(A)}{1 - \kappa(A)\varepsilon(A)} (\varepsilon(A) + \varepsilon(b))$$

Beweis: Da $Ax = b$ und $(A + \Delta A)(x + \Delta x) = b + \Delta b$ gilt

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

und daher

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

Nach Definition gilt

$$\begin{aligned} \|A^{-1}\|\|\Delta A\| &= \|A^{-1}\|\|A\|\frac{\|\Delta A\|}{\|A\|} \\ &= \kappa(A)\varepsilon(A) < 1 \end{aligned}$$

und

$$\frac{\|\Delta b\|}{\|A\|\|x\|} \leq \frac{\|\Delta b\|}{\|Ax\|} = \frac{\|\Delta b\|}{\|b\|} = \varepsilon(b)$$

also

$$\frac{\|\Delta x\|}{\|A\|\|x\|} \leq \frac{\|A^{-1}\|(\|\Delta b\|/(\|A\|\|x\|) + \varepsilon(A))}{1 - \kappa(A)\varepsilon(A)}$$

Multiplikation mit $\|A\|$ ergibt die Behauptung. □

Korollar 2.5.2 *Falls* $\varepsilon(A)\kappa(A) \ll 1$, *gilt*

$$\varepsilon(x) \stackrel{\cdot}{\leq} \kappa(A)(\varepsilon(A) + \varepsilon(b))$$

Aus diesem Grund nennt man die Größe

$$\kappa(A) = \|A\| \cdot \|A^{-1}\| \in [1, \infty]$$

die *(gewöhnliche) Kondition* der Matrix A bzgl. der Norm $\|\cdot\|$.
Eine andere Darstellung, die sich auch auf rechteckige Matrizen übertragen läßt, ist

$$\kappa(A) := \frac{\max\limits_{\|y\|=1} \|Ay\|}{\min\limits_{\|z\|=1} \|Az\|} \in [0, \infty] \tag{2.5.1}$$

Für nicht singuläre $A \in \mathrm{Mat}_n(\mathbf{R})$ stimmt dieser Ausdruck tatsächlich mit obiger Definition überein, da

$$\begin{aligned} \left(\min_{\|z\|=1} \|Az\|\right)^{-1} &= \left(\min_{z \neq 0} \frac{\|Az\|}{\|z\|}\right)^{-1} \\ &= \max_{z \neq 0} \frac{\|z\|}{\|Az\|} \quad (\text{setze } x := Az) \\ &= \max_{x \neq 0} \frac{\|A^{-1}x\|}{\|x\|} \end{aligned}$$

und daher

$$\frac{\max_{\|y\|=1} \|Ay\|}{\max_{\|z\|=1} \|Az\|} = \max_{y \neq 0} \frac{\|Ay\|}{\|y\|} \cdot \max_{x \neq 0} \frac{\|A^{-1}x\|}{\|x\|} = \|A\| \cdot \|A^{-1}\| = \kappa(A).$$

Mit dieser Formel sind die folgenden drei Eigenschaften von $\kappa(A)$ offensichtlich.

$$\begin{aligned} &\text{a)} \quad \kappa(A) \geq 1 \\ &\text{b)} \quad \kappa(\alpha A) = \kappa(A) \text{ für alle } \alpha \in \mathbf{R} \\ &\text{c)} \quad A \neq 0 \text{ singulär} \iff \kappa(A) = \infty \end{aligned} \tag{2.5.2}$$

Die letzte Eigenschaft führt uns zu der Frage:

"Wann ist A fast singulär ?"

Theoretisch ist diese Frage klar durch das Kriterium

$$\det(A) \neq 0 \Leftrightarrow A \text{ nicht singulär}$$

eindeutig für eine Matrix $A \in Mat_n(\mathbf{R})$ zu beantworten. Denken wir jedoch an unsere ersten Überlegungen zur Fehleranalyse zurück, so erweist sich diese Antwort als wenig hilfreich, da real stets eine gestörte Matrix A vorliegt und wir die Menge aller von A nicht zu unterscheidenden Matrizen, also z.B.

$$E := \{\tilde{A} = A + \Delta A \mid \|\Delta A\|/\|A\| \leq \varepsilon(A)\}$$

zu betrachten haben. In Anlehnung an die Eigenschaft c) der Kondition einer Matrix können wir definieren, was wir unter einer "numerisch singulären" Matrix verstehen wollen.

Definition 2.5.3 *Eine Matrix A heißt* fast singulär *oder* numerisch singulär, *falls*

$$\varepsilon(A)\kappa(A) \geq 1,$$

wobei $\varepsilon(A)$ die relative Genauigkeit der Matrix A ist.

Für die Rundungsfehler bei der Eingabe von A in einen Rechner nehmen wir z.B. an, daß

$$\varepsilon(A) = \text{ eps}$$

Bei experimentellen Daten ist $\varepsilon(A)$ entsprechend deren Genauigkeit meist größer anzusetzen. Interpretiert man die Definition mit Hilfe der Ungleichung (.), so zeigt sich, daß

$$\varepsilon(x) \lesssim \underbrace{\kappa(A)(\varepsilon(A) + \varepsilon(b))}_{\geq \kappa(A)\varepsilon(A) \geq 1}$$

für eine fast singuläre Matrix A. Der relative Fehler $\varepsilon(x)$ der Lösung x läßt sich nach oben nur durch eine Zahl größer oder gleich 1 abschätzen, mit anderen Worten, x kann hundertprozentig falsch sein.

Doch kommen wir zurück zu unserer eigentlichen Fragestellung der Kondition eines linearen Gleichungssystems. Muß es nicht stutzig machen, daß wir bisher nur den Begriff "Kondition einer Matrix" eingeführt haben, wo es uns doch um die *Kondition des Problems* der Lösung eines linearen Gleichungssystems $Ax = b$ mit den Eingabegrößen A *und* b geht. Tatsächlich hängt diese Kondition auch von der rechten Seite ab. In Aufgabe ? geben wir ein Beispiel für ein Gleichungssystem mit verschieden "guten" rechten Seiten.

Ein weiterer Kritikpunkt an unseren bisherigen Überlegungen ist die Fixierung auf Normen. Weitaus natürlicher wäre es, Störungen der einzelnen Komponenten zu analysieren. So wird bei der Eingabe einer Matrix A in einen Rechner jede einzelne Komponente gerundet

$$a_{ij} \to \tilde{a}_{ij} = a_{ij}(1 + \varepsilon_{ij}), \quad |\varepsilon_{ij}| \leq \text{ eps}$$

wobei Nullen exakt gespeichert werden und häufig noch mehr Struktur erhalten bleibt, wie wir bei der Analyse des Eigenwertproblems für symmetrische Matrizen gesehen haben. Auch bei experimentellen Daten können die Komponenten der Eingabegrößen A und b mit stark variierender Genauigkeit gegeben sein.

Bei neueren Untersuchungen, die zum Großteil auf Skeel zurückgehen, werden genau diese beiden Kritikpunkte beachtet.

Wir betrachten Störungen

$$\tilde{A} = A + \Delta A, \quad \tilde{b} = b + \Delta b$$

der Eingabedaten A und b mit den *komponentenweisen* Abschätzungen

$$|\Delta A| \leq \varepsilon |A|, \quad |\Delta b| \leq \varepsilon |b|,$$

wobei $|\cdot|$ der komponentenweise Betrag ist, d.h.

$$|\Delta a_{ij}| \leq \varepsilon |a_{ij}|, \quad |\Delta b_i| \text{ für alle } i, j = 1, \ldots, n$$

Die zu betrachtenden Eingabemengen sind daher

$$E_\varepsilon := \{(\tilde{A}, \tilde{b}) \mid \tilde{A} = A + \Delta A, \quad \tilde{b} = b + \Delta b \text{ mit } |\Delta A| \leq \varepsilon |A|, \quad |\Delta b| \leq \varepsilon |b|\}$$

Die Größe der Störung $\Delta A, \Delta b$ messen wir mit dem komponentenweisen relativen Fehler

$$\varepsilon(\Delta A, \Delta b) := \min\{\varepsilon \geq 0 \mid |\Delta A| \leq \varepsilon |A|, \quad |\Delta b| \leq \varepsilon |b|\}$$

Ein vernünftiges Maß für die Größe von Eingabemengen E ist daher

$$[E] := \sup_{(\tilde{A}, \tilde{b}) \in E} \varepsilon(\tilde{A} - A, \tilde{b} - b)$$

und es gilt gerade

$$[E_\varepsilon] = \varepsilon \ .$$

Auf der Seite der Resultate x messen wir die Fehler Δx durch den relativen Fehler $\|\Delta x\|_\infty / \|x\|_\infty$ bezüglich der Maximumsnorm und führen daher für Resultatmengen das Maß

$$[R] := \sup_{\tilde{x} \in R} \frac{\|\tilde{x} - x\|_\infty}{\|x\|_\infty}$$

ein. Eine Eingabemenge E_ε wird abgebildet auf die Resultatmenge

$$R_\varepsilon := \{\tilde{x} | \tilde{A}\tilde{x} = \tilde{b}, \quad (\tilde{A}, \tilde{b}) \in E_\varepsilon\}$$

Nach diesen Vorbereitungen können wir die Kondition eines linearen Gleichungssystems definieren als asymptotische Kondition des Problems

$$\kappa := \limsup_{[E]\to 0} \frac{[R]}{[E]} = \lim_{\varepsilon \to 0} \frac{[R_\varepsilon]}{[E_\varepsilon]}$$

im Sinn des ersten Abschnittes. Diese Definition deckt sich mit der folgenden von Skeel:

Definition 2.5.4 *(Skeel, 1979) Die* Kondition eines linearen Gleichungssystems $Ax = b$ *ist (mit obigen Bezeichnungen)*

$$\kappa(A, x) := \limsup_{\varepsilon(\Delta A, \Delta b))\to 0} \frac{\|\Delta x\|_\infty / \|x\|_\infty}{\varepsilon(\Delta A, \Delta b)\|x\|_\infty} = \lim_{\varepsilon \to 0} \sup_{\substack{|\Delta A| \le \varepsilon |A| \\ |\Delta b| \le \varepsilon |b|}} \frac{\|\Delta x\|_\infty}{\varepsilon \|x\|_\infty}$$

Satz 2.5.5 *(Skeel) Es gilt*

$$\kappa(A, x) = \frac{\| |A^{-1}||A||x| + |A^{-1}||b| \|_\infty}{\|x\|_\infty}$$

Beweis: Der Beweis findet sich sehr übersichtlich dargestellt in [11], Theorem 2.2 und 2.3. □

Die Überlegungen lassen sich leicht dahingehend verallgemeinern, daß man Störungen der Matrix A und der rechten Seite b getrennt betrachtet und definiert:

Definition 2.5.6 *Die Kondition des Gleichungssystems $Ax = b$ bzgl. Störungen von A ist*

$$\kappa_A(A, x) := \lim_{\varepsilon \to 0} \sup_{|\Delta A| \le \varepsilon |A|} \frac{\|\Delta a\|_\infty}{\varepsilon \|x\|_\infty}$$

Die Kondition von $Ax = b$ bzgl. Störungen von b ist

$$\kappa_b(A, x) := \lim_{\varepsilon \to 0} \sup_{|\Delta b| \le \varepsilon |b|} \frac{\|\Delta x\|_\infty}{\varepsilon \|x\|_\infty}$$

Bei der Berechnung dieser Konditionen splittet der Ausdruck für $\mathrm{cond}(A, x, b)$ gerade auf

Satz 2.5.7 *(Skeel) Es gilt*

$$a) \qquad \kappa_A(A, x) = \frac{\| |A^{-1}||A||x| \|_\infty}{\|x\|_\infty}$$

$$b) \qquad \kappa_b(A, x) = \frac{\| |A^{-1}||b| \|_\infty}{\|x\|_\infty}$$

Für den Beweis verweisen wir wieder auf [11]. Aufgrund der Ungleichung

$$\||A^{-1}||b|\|_\infty \leq \||A^{-1}||A||x|\|_\infty$$

gilt

$$\kappa_A(A,x) \leq \kappa(A,x) \leq \kappa_A(A,x) + \kappa_b(A,x) \leq 2\kappa_A(A,x)$$

Daher charakterisiert die Größe

$$\text{cond}(A,x) := \frac{\||A^{-1}||A||x|\|_\infty}{\|x\|_\infty}$$

die Kondition des linearen Gleichungssystems sowohl bzgl. Störungen von A als auch bzgl. Störungen von A und b. Die *Kondition einer Matrix* definieren wir (nach Skeel) wie folgt.

Definition 2.5.8 *Die* Kondition einer Matrix $A \in \text{Mat}_n(\mathbf{R})$ *ist*

$$\text{cond}(A) := \max_{\|x\|_\infty = 1} \text{cond}(A,x).$$

Bemerkung 2.5.9 *Mit* $e = (1,\ldots,1)$ *folgt*

$$\text{cond}(A) = \text{cond}(A,e) = \||A^{-1}||A|\|_\infty.$$

Mit diesen Größen erhalten wir eine zu Satz (2.5.7) analoge Aussage:

Satz 2.5.10 *Sei* $Ax = b$ *und* $(A + \Delta A)(x + \Delta x) = b + \Delta b$, *wobei*

$$|\Delta A| \leq \varepsilon |A|, \quad |\Delta b| \leq \varepsilon |b| \quad \textit{und} \quad \varepsilon\, \text{cond}(A) < 1.$$

Dann gilt für den relativen Fehler von x *bezüglich der Maximumsnorm, daß*

$$\frac{\|\Delta x\|_\infty}{\|x\|_\infty} \leq \varepsilon \frac{\kappa(A,x)}{1 - \varepsilon\ \text{cond}(A)} \leq 2\varepsilon \frac{\text{cond}(A,x)}{1 - \varepsilon\ \text{cond}(A)}$$

Falls $|\Delta b| = 0$, *so gilt:*

$$\frac{\|\Delta x\|_\infty}{\|x\|_\infty} \leq \frac{\text{cond}(A,x)}{1 - \varepsilon\ \text{cond}(A)}$$

Beweis: Setzt man $Ax = b$ in $(A + \Delta A)(x + \Delta x) = b + \Delta b$ ein, so folgt

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

und daher

$$\begin{aligned} |\Delta x| &\leq |A^{-1}||\Delta A|(|x| + |\Delta x|) + |A^{-1}||\Delta b| \\ &\leq \varepsilon |A^{-1}||A|(|x| + |\Delta x|) + \varepsilon |A^{-1}||b| \end{aligned}$$

Für die Maximumnorm $\|\Delta x\|_\infty$ folgt

$$\|\Delta x\|_\infty \leq \varepsilon \||A^{-1}||A||x| + |A^{-1}||b|\|_\infty + \varepsilon \||A^{-1}||A|\|_\infty \|\Delta x\|_\infty$$

also

$$\begin{aligned} \|\Delta x\|_\infty (1 - \varepsilon\ \text{cond}(A)) &\leq \varepsilon \||A^{-1}||A||x| + |A^{-1}||b|\|_\infty \\ &= \varepsilon\ \text{cond}(A,x) \cdot \|x\|_\infty \end{aligned}$$

□

Bemerkung 2.5.11 Die Skeel'sche Kondition $\mathrm{cond}(A,x)$ ist invariant unter Zeilenskalierung, d.h.

$$\mathrm{cond}(D_zA,x) = \mathrm{cond}(A,x), \quad D_z \text{ diagonal}$$

da

$$|(D_zA)^{-1}||D_zA| = |A^{-1}||D_z^{-1}||D_z||A| = |A^{-1}||A|$$

Insbesondere hat sie auch die wünschenswerte Eigenschaft, daß

$$\mathrm{cond}(D,x) = 1$$

während

$$\mathrm{cond}(D) = |d_{max}/d_{min}|$$

für eine Diagonalmatrix D mit betragsmäßig größtem bzw. kleinstem Diagonalelement d_{max} bzw. d_{min}.

Bemerkung 2.5.12 Eine weitere hervorzuhebende Eigenschaft von $\mathrm{cond}(A,x)$ ist, daß nur die Norm im Lösungsraum eingeht, da $|A^{-1}||A|$ eine Abbildung des Lösungsraums in sich ist.

2.5.2 Rückwärtsanalyse

Wir wollen nun kurz skizzieren, wie sich für das Gauß'sche Eliminationsverfahren eine Stabilitätsuntersuchung im Sinne der Rückwärtsanalyse durchführen läßt. Die Ergebnisse basieren im wesentlichen auf denen von Wilkinson in [2]. Wir gehen hier ähnlich wie bei Stewart [3] vor.

Die besprochenen Algorithmen erfordern im einzelnen nur die Auswertung von Skalarprodukten bestimmter Zeilen und Spalten von Matrizen. Die Rückwärtsanalyse liefert hier (s. (2..)) für $u,v \in \mathbf{R}^n$:

$$\mathrm{fl}(u^Tv) = (u+\delta u)^Tv = u^T(v+\delta v)$$

mit

$$|\delta u| \leq (n+2)\ \mathrm{eps}|u|$$

bzw.

$$|\delta v| \leq (n+2)\ \mathrm{eps}|v|$$

Auf der Grundlage dieser Abschätzung erhält man nach detailreicher Rechnung die folgenden Resultate:

Gestaffelte lineare Gleichungssysteme

Die gestörte Lösung $\tilde{z}$ eines Gleichungssystems

$$Lz = b$$

mit einer unteren Dreiecksmatrix L läßt sich auffassen als exakte Lösung eines gestörten Systems

$$\tilde{L}\tilde{z} = (L+\delta L)\tilde{z} = b,$$

wobei δL eine untere Dreiecksmatrix mit

$$|\delta L| \leq \beta(n+1)\,\mathrm{eps}|L| \tag{2.5.3}$$

und β eine von der benutzten Arithmetik abhängige Zahl von der Größenordnung

$$\beta = O(1)\ .$$

Komponentenweise ergibt sich anstelle von (2.5.3), daß

$$|\delta\ell_{ij}| \leq \beta \underbrace{(i-j+2)}_{\leq n+1}\,\mathrm{eps}|\ell_{ij}|,\quad i>j \tag{2.5.4}$$

Tatsächlich erweist sich diese Abschätzung in der Regel als zu pessimistisch. Häufig ist anstelle des Faktors $n+1$ in (2.5.3 eher $\sqrt{n}$ zu beobachten. Auf jeden Fall erweist sich die Auflösung eines gestaffelten linearen Gleichungssystems als im abgeschwächten Sinn stabil.

Gauß-Elimination

Die Gauß-Elimination liefert statt der "exakten" LR-Zerlegung $A = LR$ Matrizen $\tilde{L}$ und $\tilde{R}$ mit

$$\tilde{L}\tilde{R} = A + F,$$

wobei für F komponentenweise gilt, daß

$$|f_{ij}| \leq \beta \cdot n \cdot \max_k |a_{ij}^{(k)}|\,\mathrm{eps} \tag{2.5.5}$$

also

$$|F| \leq \beta \cdot n \cdot \alpha_{max}\,\mathrm{eps}\ \text{ mit }\ \alpha_{max} = \max_{i,j,k} |a_{ij}^{(k)}| \tag{2.5.6}$$

Der größte Betrag $\alpha_{\max}$ einer Komponente, welche im Laufe der Elimination in den Matrizen

$$A = A^{(1)}, A^{(2)}, \ldots, A^{(n)} = R$$

auftritt, hängt von der gewählten Pivotstrategie ab.

a) für *vollständige Pivotsuche* gilt

$$\alpha_{\max} \leq [n \cdot 2^1 \cdot 3^{\frac{1}{2}} \cdot \ldots \cdot n^{\frac{1}{n-1}}]^{1/2} \max_{i,j} |a_{ij}|$$

Diese Grenze ist nicht scharf.

b) für *Spaltenpivotsuche* gilt

$$\alpha_{\max} \leq 2^{n-1} \max_{i,j} |a_{ij}|,$$

wobei diese Grenze von dem pathologischen Beispiel von Wilkinson

$$A = \begin{bmatrix} 1 & & & 1 \\ -1 & \ddots & & \vdots \\ \vdots & \ddots & & \\ -1 & \ldots & -1 & 1 \end{bmatrix}$$

erreicht wird (siehe Aufgabe ?).

Eine feinere Rückwärtsanalyse zeigt, daß die gestörte Lösung $\tilde{x}$ aufgefaßt werden kann als exakte Lösung eines gestörten Systems

$$(A + \delta A)\tilde{x} = b$$

mit

$$|\delta A| \leq \gamma_n(2 + \gamma_n)|\tilde{L}||\tilde{R}| \quad \text{wobei} \quad \gamma_n := \frac{n \ \text{eps}}{1 - n \ \text{eps}}$$

unter der Voraussetzung, daß n eps < 1. Für einen Beweis verweisen wir auf [4]. Falls

$$|\tilde{L}||\tilde{R}| \approx |A|$$

ist die Gauß-Elimination daher im abgeschwächten Sinn stabil.

Cholesky-Zerlegung

Statt der Matrix L mit $A = LL^T$ erhält man bei der Cholesky-Zerlegung eine Matrix $\tilde{L}$ mit

$$\tilde{L}\tilde{L}^T = A + \bar{F}$$

für die die Abschätzung

$$|\bar{F}| \leq \beta n \bar{\alpha}_{\max} \text{ eps} \quad \text{mit} \quad \bar{\alpha}_{\max} := \max_{i,j} |a_{ij}| = \max_i a_{ii}$$

gilt. Die Cholesky-Zerlegung ist daher trotz des Verzichts auf eine Pivot-Strategie rückwärtsstabil (im abgeschwächten Sinn).

2.5.3 Praktische Beurteilung von Näherungslösungen

Wir stellen uns in diesem Abschnitt die Frage:

"Wie gut ist eine gegebene Näherungslösung $\hat{x}$ des Problems $Ax = b$?"

Nun ist die Tatsache, daß x exakte Lösung der linearen Gleichung $Ax = b$ ist, gleichbedeutend damit, daß das *Residuum*

$$r(x) := b - Ax$$

verschwindet, d.h. $r(x) = 0$. Es liegt daher nahe, von einer "guten" Näherungslösung $\hat{x}$ zu verlangen, daß das Residuum

$$r(\hat{x}) = b - A\hat{x}$$

möglichst "klein" ist. Die Frage ist nur, wie klein ist "klein"? Bei der Suche nach einer vernünftigen Grenze stoßen wir auf das Problem, daß die Norm $\|r(\hat{x})\|$ durch eine Zeilenskalierung beliebig verändert werden kann, obwohl das Problem selbst dadurch nicht verändert wird.

$$Ax = b \rightarrow (D_z A)x = D_z b$$

(Dies läßt sich auch an der Invarianz der Skeel'schen Kondition $\text{cond}(A)$ bezüglich Zeilenskalierung ablesen.) Das Residuum kann daher höchstens dann zur Beurteilung der Näherungslösung $\hat{x}$ herangezogen werden, wenn es eine problemspezifische Bedeutung besitzt. Besser geeignet ist das Konzept der *Rückwärtsanalyse*, wie wir es in Kapitel 2.3.2 kennengelernt haben:

Eine Näherungslösung $\hat{x}$ wird als Lösung akzeptiert, wenn $\hat{x}$ als "exakte" Lösung zu "leicht gestörten" Eingabedaten aufgefaßt werden kann, d.h.

$$(A + \Delta A)\hat{x} = b + \Delta b, \quad \text{wobei} \quad \Delta A, \Delta b \quad \text{"klein"}$$

Die inhaltliche Auffüllung der Begriffe "leicht gestört" bzw. "klein" führt wie bei der Konditionsanalyse zu zwei Konzepten: normorientiert und komponentenorientiert.

Normorientierte Beurteilung von $\hat{x}$

Wir definieren den *normweisen Rückwärtsfehler* von $\hat{x}$ durch

$$\begin{aligned} \varepsilon_N \quad := \quad & \min\{w \mid \text{es gibt } \Delta A, \Delta b, \text{ so daß } (A + \Delta A)\hat{x} = b + \Delta b \text{ und} \\ & \|\Delta A\| \leq w\|A\|\,, \ \|\Delta b\| \leq w\|b\| \,\} \end{aligned}$$

Nach Rigal und Gaches (1967) gilt dann

$$\varepsilon_N = \frac{\|r(\hat{x})\|}{\|A\|\|\hat{x}\| + \|b\|}\,.$$

Die Näherung $\hat{x}$ heißt *gut bzgl. des normweisen Rückwärtsfehlers*, falls ε_n in der Größenordnung der Eingabefehler liegt, d.h.

$$\varepsilon_N \approx \text{ Datenfehler}$$

Wie bei der Konditionsanalyse in Kapitel 2.3 werden auch hier die Nachteile des normorientierten Konzeptes deutlich. Die Normen beachten weder die möglicherweise unterschiedliche Genauigkeit der einzelnen Einträge noch die eventuell vorgegebene Besetztheitsstruktur der Matrix. Daher ist meistens das folgende Konzept vorzuziehen

Komponentenorientierte Beurteilung von $\hat{x}$

Zur komponentenorientierten Analyse geben wir eine Matrix δA und einen Vektor δb vor,

$$\delta A \geq 0, \quad \delta b \geq 0,$$

bezüglich derer wir die Störungen ΔA und Δb beurteilen wollen. Einer vorgegebenen Besetztheitsstruktur von A können wir damit dadurch Rechnung tragen, daß wir die entsprechenden Komponenten von δA Null setzen. Den *komponentenweisen Rückwärtsfehler* von $\hat{x}$ definieren wir durch

$$\begin{aligned} \varepsilon_C \quad := \quad & \min\{w \mid \text{es gibt } \Delta A, \Delta b, \text{ so daß } (A + \Delta A)\hat{x} = b + \Delta b \text{ und} \\ & |\Delta A| \leq w|\delta A|\,, \ |\Delta b| \leq w|\delta b| \,\} \end{aligned}$$

und nennen $\hat{x}$ *gut bzgl. des komponentenweisen Rückwärtsfehlers*, falls

$$\varepsilon_C(\delta A, \delta b) \approx \text{ Datenfehler}$$

Dabei geht natürlich unsere Gewichtung der Fehler durch δA und δb ein. ε_C läßt sich mit folgendem Resultat berechnen:

Satz 2.5.13 *(Prager und Oettli 1964)*

$$\varepsilon_c(\delta A, \delta b) = \max_i \frac{|r(\hat{x})|_i}{(\delta A|\hat{x}| + \delta b)_i},$$

wobei wir definieren, daß $0/0 = 0$ *und* $\xi/0 = \infty$ *für* $\xi \neq 0$.

Beweis: Wir setzen

$$\Theta := \max_i \frac{|r(\hat{x})|_i}{(\delta A|\hat{x}| + \delta b)_i}$$

und haben zu zeigen, daß $\varepsilon_C(\delta A, \delta b) = \Theta$.

a) $\varepsilon_C(\delta A, \delta b) \geq \Theta$, denn falls

$$(A + \Delta A)\hat{x} = b + \Delta b, \quad |\Delta A| \leq w\delta A, \quad |\Delta b| \leq w\delta b$$

so folgt

$$|b - A\hat{x}| = |\Delta A\hat{x} - \Delta b| \leq w\delta A|\hat{x}| + w\delta b\ .$$

b) $\varepsilon_C(\delta A, \delta b) \leq \Theta$: wir können das Residuum $r = b - a\hat{x}$ schreiben als

$$r = D(\delta A|\hat{x}| + \delta b),$$

wobei D die Diagonalmatrix $D = \text{diag}\,(d_1, \ldots, d_n)$ mit den Diagonalelementen

$$d_i = \frac{r_i}{(\delta A|\hat{x}| + \delta b)}$$

ist. Da $\Theta = \max_i |d_i|$, gilt $|D| \leq \Theta I$. Setzen wir nun

$$\begin{aligned} \Delta A &:= D\delta A\ \text{diag}\,(\text{sgn}(\hat{x})) \\ \Delta b &:= -D\delta b, \end{aligned}$$

so gilt $|\Delta A| \leq \Theta\delta A$, $|\Delta b| \leq \Theta\delta b$ und

$$\begin{aligned} (A + \Delta A)\hat{x} - (b + \Delta b) &= A\hat{x} + D\delta A|\hat{x}| - b + D\delta b \\ &= r - D(\delta A|\hat{x}| + \delta b) \\ &= 0 \end{aligned}$$

□

Sind alle Einträge mit dem gleichen relativen Fehler δ behaftet, z.B. dem Rundungsfehler $\delta = \text{eps}$, so setzt man

$$\begin{aligned} \delta A &:= |A| \\ \delta b &:= |b| \end{aligned}$$

Für den zugehörigen Rückwärtsfehler folgt

$$\varepsilon := \varepsilon_c(|A|, |b|) = \max_i \frac{|r(\hat{x})|_i}{(|A||\hat{x}| + |b|)_i}$$

(Dieser Ausdruck ist wieder invariant gegenüber Zeilenskalierung). Im Fall von Rundungsfehlern ist die Lösung daher akzeptabel, falls

$$\varepsilon \leq O(n)\ \text{eps}$$

2.5.4 Nachiteration

Haben wir einmal festgestellt. daß die Näherungslösung $\tilde{x}$ von $Ax = b$ nicht akzeptabel ist, bleibt die Frage, wie wir zu einer besseren Lösung gelangen können. Natürlich können wir einfach die Lösung $\tilde{x}$ verwerfen und mit höherer Genauigkeit eine neue Lösung berechnen. Dabei gehen jedoch alle Informationen, die wir bei der Berechnung von $\tilde{x}$ erarbeitet haben, verloren. Dies wird bei der *iterativen Verbesserung* oder *Nachiteration* (engl.: iterative refinement) vermieden.

Gehen wir aus von einer beliebigen Methode, mit der sich näherungsweise eine Lösung $\tilde{x}$ linearen Gleichungssystems $Ax = b$ bestimmen läßt, und beschreiben diese Methode mit einer Matrix $\tilde{A}$, so daß

$$\tilde{A}\tilde{x} = b, \quad \tilde{A} = A + F, \quad F \neq 0$$

Verwenden wir beispielsweise die Gauß-Elimination, so ist

$$\tilde{A} = LR$$

Für das Residuum der ersten Näherungslösung $x^0 := \tilde{x}$ gilt

$$r(x^0) = b - Ax^0 = A(x - x^0) \neq 0$$

und der Fehler

$$\Delta x^0 := x - x^0$$

ist gerade die "exakte" Lösung von

$$A\Delta x^0 = r(x^0) \tag{2.5.7}$$

Die Idee der Nachiteration besteht nun darin, die Korrektur Δx^0 mit Hilfe der gleichen Methode x_0 aus (2.5.7) zu berechnen und diesen Prozeß zu iterieren. Wir kommen so zu dem Verfahren ($i = 0, 1, \ldots$)

$$\begin{aligned} &\text{a)} \quad \tilde{A}x^0 &&= b \\ &\text{b)} \quad \tilde{A}\Delta x^i &&= r(x^i) \\ &\text{c)} \quad x^{i+1} &&= x^i + \Delta x^i \end{aligned} \tag{2.5.8}$$

Daß dies funktioniert, bestätigt der folgende Satz.

Satz 2.5.14 *Mit den obigen Bezeichnungen gelte (ρ = Spektralradius)*

$$\rho(\tilde{A}^{-1}F) \leq \Theta < 1$$

Dann konvergiert die Folge $\{x^i\}$ gegen die "exakte" Lösung x von $Ax = b$.

Beweis: siehe Aufgabe 2.14 □

Dieser Satz ist zwar theoretisch ganz beruhigend, praktisch jedoch kaum zu gebrauchen. Zum einen wissen wir, daß es gar nicht sinnvoll ist, von einer "exakten" Lösung auszugehen (da A und b fehlerbehaftet sind), zum anderen gibt uns dieser Satz kein Kriterium an, wann die Iteration vernünftigerweise abzubrechen ist. Für praktische Zwecke ist der folgende Satz wesentlich besser geeignet.

Satz 2.5.15 *(Jankowski/Woźniakowski 1977, s. [13]) Sei ein numerisches Verfahren zur Lösung von $Ax = b$ gegeben, für das der relative Fehler durch*

$$\frac{\|\tilde{x} - x\|}{\|x\|} \leq \varepsilon_x < 1$$

abschätzen läßt. Gilt zusätzlich

$$\varepsilon_x \kappa(A) = O(1), \tag{2.5.9}$$

so erzeugt die Nachiteration nach endlich vielen Schritten eine Lösung x^* mit dem normweisen Rückwärtsfehler

$$\varepsilon_N \leq K_n \text{ eps},$$

wobei K_n in typischen Fällen von der Ordnung $K_n = O(n)$ ist.

Beweis: siehe BIT 17, 303 - 311 (1977) □

Korollar 2.5.16 *Nachiteration macht einen vorwärtsstabilen Algorithmus rückwärtsstabil, falls*

$$O(n)\varepsilon(A)\kappa(A)^2 \approx 1$$

Beweis: Wir gehen davon aus, daß $\varepsilon(b) = 0$ und

$$\varepsilon(A)\kappa(A) \ll 1$$

Dann folgt nach Korollar 2.5.2, daß

$$\varepsilon(x) \approx \varepsilon(A)\kappa(A) \ll 1$$

und für einen vorwärtsstabilen Algorithmus

$$\varepsilon_x = O(n)\varepsilon(x) = O(n)\varepsilon(A)\kappa(A)$$

Eingesetzt in (2.5.9) folgt die Behauptung. □

Diese Ergebnisse bezogen sich auf die normweise Rückwärtsanalyse. Bezüglich der komponentenweisen Rückwärtsanalyse hat Skeel gezeigt, daß nur eine Nachiteration bei der Gauß-Elimination die komponentenweise Rückwärtsstabilität impliziert.

Satz 2.5.17 *(Skeel 1980) Falls*

$$K_n \operatorname{cond}(A^{-1})\ \sigma(A, x) \text{ eps} < 1,$$

wobei $\operatorname{cond}(A^{-1}) = \||A||A^{-1}|\|_\infty$ *die Skeel'sche Kondition und*

$$\sigma(A, x) := \frac{\max_i(|A||x|)_i}{\min_i(|A||x|)_i}$$

und K_n eine Konstante von der Ordnung $K_n = O(n)$, so liefert die Gauß-Elimination mit Spaltenpivotsuche und einer Nachiteration einen komponentenweisen Rückwärtsfehler

$$\varepsilon := \varepsilon_c(|A|, |b|) \ \dot{\leq}\ (n+1) \text{ eps}$$

Beweis: Skeel, Math. Comp. 35, 817 - 832 (1980) □

Bemerkung 2.5.18 Eine nette Zugabe der Nachiteration ist eine ungefähre Konditionsbestimmung durch

$$\kappa(A) \doteq \frac{\|\delta x_{fin}\|_{skal}}{\text{eps}} \doteq \frac{\varepsilon_{fin}(x)}{\varepsilon(A)},$$

wobei δx_{fin} die Korrektur der letzten Nachiteration.

2.6 Übungen

Aufgabe 2.1 Die Nullstellenbestimmung von Polynomen in Koeffizientendarstellung ist i. a. ein schlecht konditioniertes Problem. Zur Illustration betrachte man das folgende Polynom:

$$P(x) = x^4 - 4x^3 + 6x^2 - 4x + a, \quad a = 1$$

Wie ändern sich die Nullstellen x_i von P, wenn der Koeffizient a zu $\bar{a} := a - \epsilon$, $0 < \epsilon \leq$ eps, verfälscht wird? Man gebe die Kondition des Problems an. Auf wieviele Stellen genau ist die Lösung bei einer relativen Maschinengenauigkeit von eps $= 10^{-16}$ berechenbar?

Aufgabe 2.2 Man bestimme

$$\alpha = \left(\frac{1+\frac{1}{2n}}{1-\frac{1}{2n}}\right)^n - e\left(1 + \frac{1}{12n^2}\right)$$

für $n = 10^6$ auf drei Stellen genau.

Hinweis: $\alpha > 0$

Aufgabe 2.3 Gesucht sei die Nullstelle des kubischen Polynoms

$$z^3 + 3qz - 2r = 0, \quad r, q > 0$$

Nach Cardano ist ein reelle Wurzel gegeben durch

$$z = \left(r + \sqrt{q^3 + r^2}\right)^{\frac{1}{3}} + \left(r - \sqrt{q^3 + r^2}\right)^{\frac{1}{3}}$$

Diese Formel ist numerisch ungünstig, da zwei Kubikwurzeln berechnet werden müssen und für $r \to 0$ Totalauslöschung auftritt. Man gebe eine auslöschungsfreie Form für z an, welche nur die Berechnung einer Kubik- und einer Quadratwurzel erfordert.

Aufgabe 2.4 Berechnen Sie die Kondition der Auswertung eines durch die Koeffizienten $a_0, \ldots, a_n$ gegebenen Polynoms

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

an der Stelle x zum einen bezüglich Störungen $a_i \to \tilde{a}_i = a_i(1+\epsilon_i)$ der Koeffizienten und zum anderen bezüglich Störungen $x \to \tilde{x} = x(1+\epsilon)$ von x. Betrachten Sie insbesondere das Polynom

$$p(x) = 8118x^4 - 11482x^3 + x^2 + 5741x - 2030$$

an der Stelle $x = 0.707107$. Das "exakte" Resultat ist

$$p(x) = -1.9152732527082 \cdot 10^{-11}$$

Ein Rechner liefere

$$\tilde{p}(x) = -1.9781509763561 \cdot 10^{-11}$$

Beurteilen Sie die Lösung anhand der Kondition des Problems.

Aufgabe 2.5 Rechnen Sie die Formel aus der Vorlesung

$$\tilde{s}_n = \sum_{i=1}^{n} a_i b_i (1+\kappa_i), \quad \kappa_i = (1+\epsilon_i)(1+\delta_i)(1+\mu_i)(1+\alpha_i)\cdots(1+\alpha_n) - 1$$

für die Stabilitätsanalyse des Skalarproduktalgorithmus nach.

Aufgabe 2.6 Seien $\|\cdot\|_1$ und $\|\cdot\|_2$ Normen auf dem $\mathbf{R}^n$ bzw. $\mathbf{R}^m$. Zeigen Sie, daß durch

$$\|A\| := \sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_1}$$

eine Norm auf dem Raum $\mathrm{Mat}_{m,n}(\mathbf{R})$ der reellen (m,n) Matrizen definiert wird.

Aufgabe 2.7 Die p-Normen auf dem $\mathbf{R}^n$ sind für $1 \leq p < \infty$ definiert durch

$$\|x\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

und für $p = \infty$ durch

$$\|x\|_\infty := \max_{i=1,\ldots,n} |x_i|$$

Zeigen Sie für die zugeordneten Matrixnormen

$$\|A\|_p := \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$$

die folgenden Aussagen:

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

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

(c) $\|A\|_2 \leq \sqrt{\|A\|_1 \|A\|_\infty}$

(d) $\|AB\|_p \leq \|A\|_p \|B\|_p$ für $1 \leq p \leq \infty$

Aufgabe 2.8 Gegeben sei die Matrix

$$A_\epsilon = \begin{pmatrix} 1 & 1 \\ 1 & 1+\epsilon \end{pmatrix}$$

und die beiden rechten Seiten

$$b_1^T = (1,1), \quad b_2^T = (-1,1)$$

Berechnen Sie die beiden Lösungen x_1 und x_2 der Gleichungen $Ax_i = b_i$, die Kondition $\kappa_\infty(A_\epsilon)$ und die Skeel'sche Konditionen $\mathrm{cond}(A_\epsilon, x_1)$ und $\mathrm{cond}(A_\epsilon, x_2)$. Wie sind diese Ergebnisse zu interpretieren?

Aufgabe 2.9 Sei A eine nicht singuläre Matrix, die durch eine kleine Störung δA zu $\tilde{A} := A + \delta A$ verfälscht werde, und $\|\cdot\|$ eine submultiplikative Norm mit $\|I\| = 1$. Zeigen Sie:

a) Falls $\|B\| < 1$, so existiert $(I - B)^{-1} = \sum_{k=0}^{\infty} B^k$ und es gilt

$$\|(I - B)^{-1}\| \leq \frac{1}{1 - \|B\|}$$

b) Falls $\|A^{-1}\delta A\| \leq \epsilon < 1$, so gilt

$$\kappa(\tilde{A}) \leq \frac{1 + \epsilon}{1 - \epsilon}\kappa(A)$$

Aufgabe 2.10 Sei A eine Matrix mit den Spalten A_j, $j = 1, \ldots, n$. Wir definieren zwei Matrixnormen durch

$$\begin{aligned} \|A\|_{\square} &:= \max_j \|A_j\|_2 \\ \|A\|_F &:= \sqrt{\sum_{i,j} a_{ij}^2} \quad \text{(Frobenius-Norm)} \end{aligned}$$

Zeigen Sie:

a) Durch $\|\cdot\|_{\square}$ und $\|\cdot\|_F$ werden Normen auf $\mathrm{Mat}_{m,n}(\mathbf{R})$ definiert.

b) $\|Ax\|_2 \leq \|A\|_{\square}\|x\|_1$ und $\|A^T x\|_\infty \leq \|A\|_{\square}\|x\|_2$

c) $\|A\|_F \leq \sqrt{n}\|A\|_{\square}$ und $\|AB\|_{\square} \leq \|A\|_F\|B\|_{\square}$

Aufgabe 2.11 Es sei für $x \in \mathbf{R}$:

$$s_k := \sin(kx), \quad c_k := cos(kx), \quad dc_k := c_{k+1} - c_k$$

Mit Hilfe der Formeln

$$(A) \quad \begin{cases} c_0 := 1, \quad c_1 := \cos(x) \\ c_{k+1} := 2c_1c_k - c_{k-1} \quad \text{für } k = 1, \ldots, n-1 \end{cases}$$

$$(B) \quad \begin{cases} c_0 := 1, \quad c_1 := \cos(x), \quad s_0 := 0, \quad s_1 := \sin(x) \\ c_{k+1} := c_1c_k - s_1s_k \\ s_{k+1} := s_1c_k + c_1s_k \end{cases} \quad \text{für } k = 1, \ldots, n-1$$

$$(C) \quad \begin{cases} c_0 := 1, \quad dc_0 := -2\sin^2(x/2) \\ c_k := c_{k-1} + dc_{k-1} \\ dc_k := -4\sin^2(x/2)c_k + dc_{k-1} \end{cases} \quad \text{für } k = 1, \ldots, n-1$$

kann man $\cos(nx)$ rekursiv berechnen. Schreiben Sie ein Programm, das $\cos(x)$ durch Auswertung der Standardfunktion und $\cos(nx)$ mit Hilfe von (A), (B) bzw. (C) berechnet und vergleichen Sie die Ergebnisse für $n = 1000$ und verschiedene x. Wiederholen Sie die Rechnungen mit gestörten Eingabegrößen c_1, s_1 und dc_0 mit einer relativen Störung von $\epsilon = 1 \cdot 10^{-4}$.

Aufgabe 2.12 Gegeben sei das lineare Gleichungssystem $Ax = b$ mit

$$A = \begin{pmatrix} 0.780 & 0.563 \\ 0.913 & 0.659 \end{pmatrix}, \quad b = \begin{pmatrix} 0.217 \\ 0.254 \end{pmatrix}$$

auf die relative Genauigkeit $\text{eps} = 10^{-4}$. Beurteilen Sie mit Hilfe des Satzes von Prager und Oettli die Genauigkeit der beiden Näherungslösungen

$$\tilde{x}_1 = \begin{pmatrix} 0.999 \\ -1.001 \end{pmatrix} \quad \text{und} \quad \tilde{x}_2 = \begin{pmatrix} 0.341 \\ -0.087 \end{pmatrix}.$$

Aufgabe 2.13 Zeigen Sie, daß sich bei dem pathologischen Beispiel von Wilkinson

$$A = \begin{bmatrix} 1 & & & 1 \\ -1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ -1 & \cdots & -1 & 1 \end{bmatrix}$$

bei Spaltenpivotsuche für den Betrag des maximalen Pivotelements

$$|\alpha_{\max}| = 2^{n-1}$$

ergibt.

Aufgabe 2.14 Zeigen Sie, daß das Verfahren zur iterativen Nachbesserung einer Näherungslösung x_0 des linearen Gleichungssystems $Ax = b$

$$(A) \qquad x_0 := \text{ Lösung von } LRx_0 = b, \text{ wobei } LR = A + F \text{ mit } F \neq 0$$

$$(B) \qquad \begin{cases} r_i := Ax_i - b \\ \delta x_i := \text{ Lösung von } LR\delta x_i = -r_i \qquad \text{für } i = 1, 2, \ldots \\ x_{i+1} := x_i + \delta x_i \end{cases}$$

konvergiert, falls

$$\|(LR)^{-1}F\|_p \leq \theta < 1.$$

3 Orthogonalisierungsverfahren

Jeder Eliminationsprozeß für lineare Gleichungssysteme läßt sich formal in der Form

$$A = A^{(1)} \to A^{(2)} \to \cdots \to A^{(k)} = R$$

darstellen. Wenden wir darauf die differentielle Fehleranalyse an, so summieren sich (in linearisierter Theorie) die Zwischenfehler gemäß

$$\varepsilon(x) \doteq \sum_{j=1}^{k} \varepsilon_j(x) \quad \text{mit} \quad \varepsilon_j(x) = \varepsilon(A^{(j)}) \operatorname{cond}(A^{(j)})$$

Beim Gauß-Algorithmus ist trotz Pivotstrategie nicht auszuschließen, daß

$$\operatorname{cond}(A^{(j)}) \gg \operatorname{cond}(A) \quad \text{für ein } j > 1$$

(In der Rückwärtsanalyse würde sich dies durch Anwachsen von α_{max} bemerkbar machen).

Um solche Verschlechterungen der Kondition in Zwischenschritten (besonders bei ohnehin schlecht konditionierten Systemen) zu verhindern, wählt man orthogonale Transformationen Q_j zur Elimination:

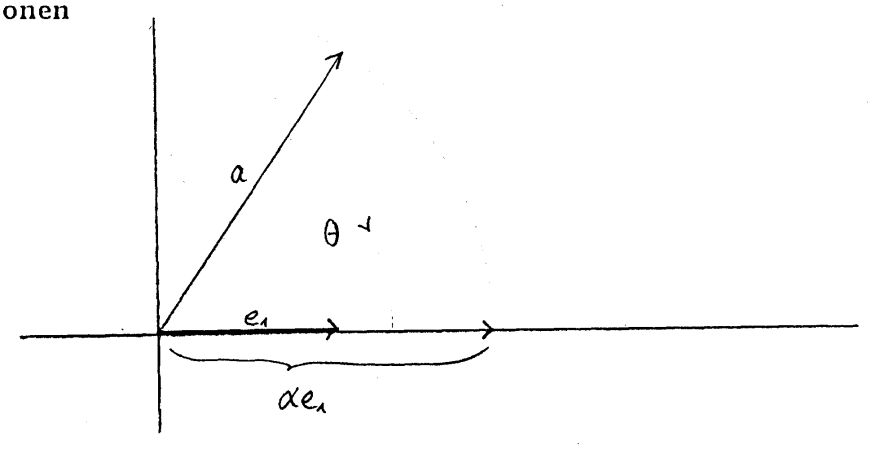
$$A^{(j+1)} = Q_j A^{(j)} \quad , \quad Q_j \in O(n)$$

Dann gilt wegen $\|Qx\|_2 = \|x\|_2$, daß

$$\operatorname{cond}_2(A^{(j)}) = \operatorname{cond}_2(A)$$

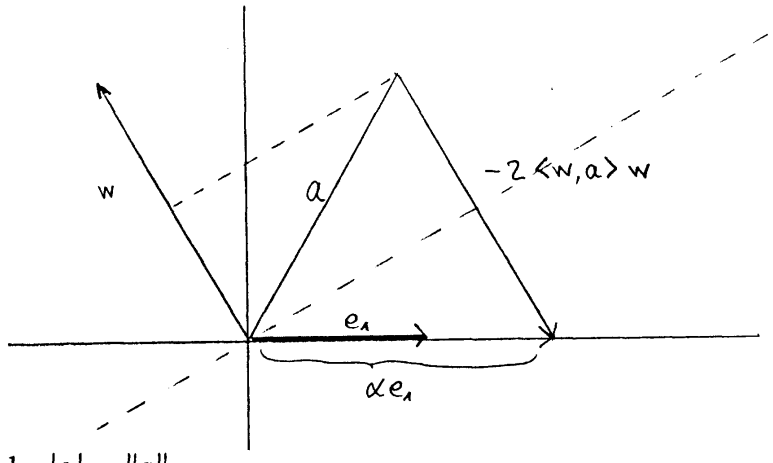
Solche Verfahren zeichnen sich durch eine inhärente Stabilität aus, sind aber, wie wir sehen werden, auch etwas teurer als z.B. die Gauß-Elimination. Die in Frage kommenden orthogonalen Transformationen lassen sich für $n = 2$ leicht geometrisch ableiten (a soll auf αe_1 abgebildet werden).

a) **Rotationen**



1. $|\alpha| = \|a\|_2$
2. $\alpha e_1 = Qa$ mit $Q = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$

b) Reflexionen



1. $|\alpha| = \|a\|_2$
2. $\alpha e_1 = a - 2 < w, a > w = a - 2ww^T a$
3. (Richtung von w) $= a - \alpha e_1$

3.1 Givens-Rotationen

Als *Givens-Rotationen* (GIVENS 1953) bezeichnet man Matrizen der Form:

$$\Omega_{k\ell} := \begin{bmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & c & & s & & & \leftarrow k \\ & & & & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & & \\ & & & -s & & c & & & \leftarrow \ell \\ & & & & & & & 1 & \\ & & & & & & & & \ddots \\ & & & & & & & & & 1 \end{bmatrix} \in Mat_n(\mathbf{R})$$

mit $c^2 + s^2 = 1$. Dabei sollen c und s natürlich an $\cos\theta$ und $\sin\theta$ erinnern. Geometrisch beschreibt die Matrix eine Drehung um den Winkel θ in der (k, ℓ)-Ebene.

Anwendung auf Vektoren

Explizit erhalten wir bei der Anwendung von $\Omega_{k\ell}$ auf einen Vektor $x \in \mathbf{R}^n$

$$y = \Omega_{k\ell} x \implies \begin{cases} y_2 &= x_i & \text{für } i \neq k, \ell \\ y_k &= cx_k + sx_\ell & \\ y_\ell &= -sx_k + cx_\ell & \end{cases} \tag{3.1.1}$$

Multiplizieren wir eine Matrix

$$A = [A_1, \ldots, A_n]$$

mit den Spalten $A_1, \ldots, A_n$ von links mit $\Omega_{k\ell}$, so operiert die Givens-Rotation auf den Spalten, d.h.

$$\Omega_{k\ell} A = [\Omega_{k\ell} A_1, \ldots, \Omega_{k\ell} A_n] \,,$$

es werden daher wegen (.) nur die beiden Zeilen k und ℓ der Matrix A verändert. Dies ist insbesondere wichtig, wenn man bei der Transformation Besetztheitsstrukturen der Matrix erhalten möchte.

Wie sind nun die Koeffizieten c und s zu bestimmen, um eine Komponente a_ℓ des Vektors x zu eliminieren? Da $\Omega_{k\ell}$ nur auf der (k,ℓ)-Ebene operiert, genügt es, das Prinzip an dem Fall $n = 2$ zu erläutern:

$$\begin{aligned} & \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix}, \quad a^2 + b^2 \neq 0 \\ \iff \quad & ca + sb = r \quad \text{und} \quad -sa + cb = 0 \\ \iff \quad & r = \pm\sqrt{a^2 + b^2}, \quad c = a/r, \quad s = b/r \end{aligned}$$

Um x_ℓ zu eliminieren, d.h. $y_\ell := 0$, setzen wir daher

$$r := \sqrt{x_k^2 + x_\ell^2} \,, \; c := x_k/r \,, \; s := x_\ell/r$$

Um Exponentenüberlauf zu vermeiden, berechnet man c und s häufig mit der billigeren Formel (5 flops + 1 sqrt)

$$\begin{aligned} \tau &:= x_k/x_\ell \\ s &:= 1/\sqrt{1+\tau^2} \\ c &:= s \cdot \tau \end{aligned}$$

falls $|x_\ell| > |x_k|$ und analog, falls $|x_k| \geq |x_\ell|$.

QR-Zerlegung einer Matrix

Um A mit Hilfe von Givens-Rotationen auf obere Dreiecksgestalt zu transformieren, eliminiert man Spalte für Spalte die von Null verschiedenen Matrix-Komponenten von der letzten Zeile bis zur Diagonale. Am Beispiel einer vollbesetzten $(4,4)$-Matrix läßt sich der Algorithmus wie folgt veranschaulichen:

$$A = \begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix} \xrightarrow{(4,3)} \begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ 0 & x & x & x \end{bmatrix}$$

$$\begin{matrix}(3,2)\\ \longrightarrow\end{matrix} \quad \cdots \quad \begin{matrix}(2,1)\\ \longrightarrow\end{matrix} \begin{bmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{bmatrix} \begin{matrix}(4,3)\\ \longrightarrow\end{matrix} \begin{bmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \end{bmatrix}$$

$$\begin{matrix}(3,2)\\ \longrightarrow\end{matrix} \quad \cdots \quad \begin{matrix}(4,3)\\ \longrightarrow\end{matrix} \begin{bmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \end{bmatrix}$$

Dabei müssen wir bei vollbesetzter Ausgangsmatrix A für die QR- Zerlegung

$$\sim \tfrac{1}{2}n^2 \quad \text{sqrt-Auswertungen und}$$
$$\sim \tfrac{4}{3}n^3 \quad \text{Multiplikationen}$$

aufwenden. Der Vergleich mit der Gauß-Elimination ($\sim n^3/3$ Multiplikationen) zeigt, daß die größere Stabilität mit einem erheblich höheren Aufwand erkauft werden muß. Zu beachten ist jedoch, daß der Vergleich für dünnbesetzte Matrizen wesentlich günstiger ausfällt. So benötigt man nur $n-1$ Givens-Rotationen, um eine sogenannte *Hessenberg-Matrix*

$$A = \begin{bmatrix} * & \cdots & \cdots & \cdots & * \\ * & \ddots & & & \vdots \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & * & * \end{bmatrix}$$

die fast schon eine obere Dreiecksmatrix ist und nur in der ersten Nebendiagonale weitere von Null verschwindende Komponenten besitzt, auf Dreiecksgestalt zu bringen.

Bemerkung 3.1.1 Es gibt eine etwas günstigere Variante der Givens-Transformationen. Speichert man A mit einer Zeilenskalierung $D_s A$, so lassen sich die Givens-Rotationen (ähnlich wie bei der rationalen Cholesky-Zerlegung) ohne sqrt-Auswertungen realisieren. Diese Art der QR-Zerlegung wurde von GENTLEMAN und HAMMARLING 1973 entwickelt und heißt *fast Givens* oder auch *rationaler Givens*. Die Zerlegung ist sogar in variant gegen Spaltenskalierung, d. h.

$$A = QR \quad \Rightarrow \quad AD = Q(RD) \text{ für eine Diagonalmatrix } D\,.$$

3.2 Householder-Reflexionen

Matrizen $Q \in Mat_n(\mathbf{R})$ der Form

$$Q = I - 2\frac{vv^T}{v^T v} \quad , \quad v \in \mathbf{R}^n$$

oder

$$Q = I - 2ww^T \quad , \quad w \in \mathbf{R}^n \text{ mit } \|w\|_2 = 1$$

heißen *Householder-Reflexionen.* Sie beschreiben gerade die Reflexion der auf v bzw. w senkrecht stehenden Ebene $\text{span}(v)^\perp$ bzw. $\text{span}(w)^\perp$. Insbesondere hängt Q nur von der Richtung $\text{span}(v)$ von v ab.

Den Nachweis der folgenden Eigenschaften von Householder-Reflexionen lassen wir als einfache Übung

a) $Q^T = Q$, d.h. Q ist symmetrisch

b) $Q^TQ = I$, d.h. Q ist orthogonal

c) $Q^2 = I$, d.h. Q ist involutorisch

Anwendung auf Vektoren

Wenden wir Q auf einen Vektor $y \in \mathbf{R}^n$ an, so gilt

$$z := Qy = (I - \frac{2vv^T}{v^Tv})y = y - \frac{2v^Ty}{v^Tv}v$$

Soll y auf ein Vielfaches αe_1 des ersten Einheitsvektors e_1 abgebildet werden, d.h.

$$z = \alpha e_1 = y - \frac{2v^Ty}{v^Tv}v \in \text{span}(e_1)$$

so folgt

a) $|\alpha| = \|z\|_2 = \|y\|_2$

b) $v \in \text{span}(y - z) = \text{span}(y - \alpha e_1)$

und wir können Q alternativ bestimmen durch

a) $v := y - \alpha e_1$ mit $\alpha = \pm\|y\|_2$

b) $w := \dfrac{y - \alpha e_1}{\|y - \alpha e_1\|_2}$ mit $\alpha = \pm\|y\|_2$

Bei der Realisierung sind die beiden folgenden Punkte zu beachten:

1. Um Auslöschung bei der Berechnung von

$$v = y - \alpha e_1 = \begin{pmatrix} y_1 - \alpha \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

zu vermeiden, wähle

$$\alpha := -\text{sgn}(y_1)\|y\|_2$$

2. Für beliebige $x \in \mathbf{R}^n$ berechnet sich Qx am einfachsten durch

$$Qx = x - \frac{2v^T x}{v^T v} x = x + \frac{v^T x}{\alpha < v, e_1 >} ,$$

wobei $< v, e_1 >= y_1 - \alpha$ die erste Komponente von v ist. Dies folgt aus der Tatsache, daß

$$\begin{aligned} v^T v &= < y - \alpha e_1, y - \alpha e_1 > \\ &= \|y\|_2^2 - 2\alpha < y, e_1 > + \alpha^2 \\ &= -2\alpha(y_1 - \alpha) . \end{aligned}$$

QR-Zerlegung einer Matrix

Mit Hilfe der Householder-Reflexionen können wir nun sukzessive die Subdiagonalelemente einer Matrix

$$A = [A_1, \ldots, A_n]$$

eliminieren. Für den ersten Schritt erhalten wir

$$A = A^{(1)} \to A^{(2)} := Q_1 A^{(1)} = \begin{bmatrix} \alpha_1 & & \\ 0 & & \\ \vdots & A_2^{(2)} \cdots A_n^{(2)} & \\ 0 & & \end{bmatrix} ,$$

wobei

$$Q_1 = I - 2\frac{v_1 v_1^T}{v_1^T v_1}$$

mit

$$\begin{aligned} v_1 &:= A_1 - \alpha_1 e_1 \\ \alpha_1 &:= -\mathrm{sgn}(a_{11}) \|A_1\|_2 \end{aligned}$$

Durch die wiederholte Anwendung auf die $(n-k+1, n-k+1)$-Restmatrizen $B^{(k)}$

$$A^{(k)} = \begin{bmatrix} * & \cdots & & & * \\ & \ddots & & & \vdots \\ & & * & \cdots & * \\ & & 0 & & \\ & & \vdots & B^{(k)} & \\ & & 0 & & \end{bmatrix}$$

mit orthogonalen Matrizen

$$Q_k = \left[\begin{array}{c|c} I_{k-1} & 0 \\ \hline 0 & \bar{Q}_k , \end{array}\right]$$

wobei $\bar{Q}_k$ wie im ersten Schritt mit $B^{(k)}$ anstelle von $A^{(1)}$ gebildet wird, erhalten wir die Zerlegung

a) $R = A^{(n)} = Q_{n-1} \cdots Q_1 A$

b) $A = QR$ mit $Q = Q_1 \cdots Q_{n-1}$.

Wollen wir damit ein lineares Gleichungssystem der Form $Ax = b$ lösen, so müssen wir wie bei der Gauß-Elimination drei Schritte durchführen.

1) $A = QR$, QR-Zerlegung

2) $c = Q^T b$, Transformation der rechten Seite

3) $Rx = c$, Auflösung des gestaffelten Systems

Der Aufwand für die QR-Zerlegung beträgt

$$2 \cdot \sum_{k=1}^{n-1} (n+1-k)^2 \sim \frac{2}{3} n^3 \text{ Multiplikationen und } n-1 \text{ sqrt-Auswertungen}$$

Bei der Implementierung auf einem Rechner werden die Operationen meist innerhalb der Matrix A durchgeführt. Neben der oberen Dreiecksmatrix R müssen auch die Householder-Vektoren $v_1, \ldots, v_{n-1}$ abgespeichert werden. Dazu speichert man die Diagonalelemente

$$\begin{aligned} r_{ii} &= \alpha_1 \text{ für } i = 1, \ldots, n-1 \\ r_{nn} &= a_{nn} \end{aligned}$$

getrennt in einem Vektor, so daß die $v_1, \ldots, v_{n-1}$ in der unteren Hälfte von A Platz finden.

Eine andere Möglichkeit ist, die Householder-Vektoren so zu normieren, daß die erste Komponente $< v_i, e_i >$ jeweils 1 ist und nicht abgespeichert werden muß.

Ähnlich wie bei der Gauß-Elimination gibt es auch bei der QR-Zerlegung eine Vertauschungsstrategie, die *Spaltentauschstrategie* von BUSINGER und GOLUB (1965). Anders als bei der Gauß-Elimination ist diese Strategie jedoch für die numerische Stabilität des Algorithmus nebensächlich. Man tauscht die Spalte mit maximaler 2-Norm nach vorne, so daß nach dem Tausch gilt:

$$\|A_1\|_2 = \max_j \|A_j\|_2 =: \|A\|_\square$$

Mit dieser Strategie gilt dann

$$r_{i+1,i+1} \leq r_{ii}$$

für die Diagonalelemente r_{ii} von R.

Als eine der Kondition verwandte Größe erweist sich die *Subkondition* von Deuflhard und Sautter (1979)

$$\mathrm{sc}(A) := \frac{|r_{11}|}{|r_{nn}|}$$

Es gilt analog zu den Eigenschaften der Kondition cond_p

a) $\mathrm{sc}(A) \geq 1$

b) $\mathrm{sc}(\alpha A) = \mathrm{sc}(A)$

c) $A \neq 0$ singulär $\iff \mathrm{sc}(A) = \infty$

d) $\mathrm{sc}(A) \leq \mathrm{cond}_2(A)$ (daher der Name)

Die Subkondition kann auch herangezogen werden, um zu entscheiden, ob A nicht schon als singulär anzusehen ist, indem wir definieren

Definition 3.2.1 *Eine Matrix A heißt* fast singulär *bei einer QR-Zerlegung mit Spaltentauschstrategie, falls*

$$\varepsilon(A)\,\mathrm{sc}(A) \geq 1$$

Daß dieser Begriff überhaupt sinnvoll ist, kann mit einer detaillierten Fehleranalyse begründet werden.

3.3 Lineare Ausgleichsrechnung

In diesem Paragraphen wollen wir uns mit einer weiteren Methode beschäftigen, die Gauß bei der Berechnung der Bahn des Planetoiden Ceres entwickelt hat. Sie ist auch unter dem Namen

- Gauß'sche Methode der kleinsten Fehlerquadrate
- lineare Regressionsanalyse
- linear least squares
- maximum-likelihood-Methode

bekannt.

3.3.1 Problemstellung

Wir gehen aus von folgender Problemstellung:
Gegeben seien m Meßpunkte

$$(t_i, y_i), \quad t_i, y_i \in \mathbf{R}, \quad i = 1, \ldots, m$$

die zum Beispiel den Zustand y_i eines Objektes zur Zeit t_i beschreiben. Man nimmt an, daß diesen Messungen eine Gesetzmäßigkeit zugrunde liegt, so daß sich die Abhängigkeit von y von t durch eine *Modellfunktion* φ mit

$$y(t) = \varphi(t; x_1, \ldots, x_n)$$

ausdrücken läßt, wobei in die Modellfunktion n unbekannte Parameter $x_1, \ldots, x_n$ eingehen.

Beispiel 3.3.1 Ohm'sches Gesetz $y = xt = \varphi(t; x)$, wobei

t : Stromstärke

y : Spannung

x : Ohm'scher Widerstand

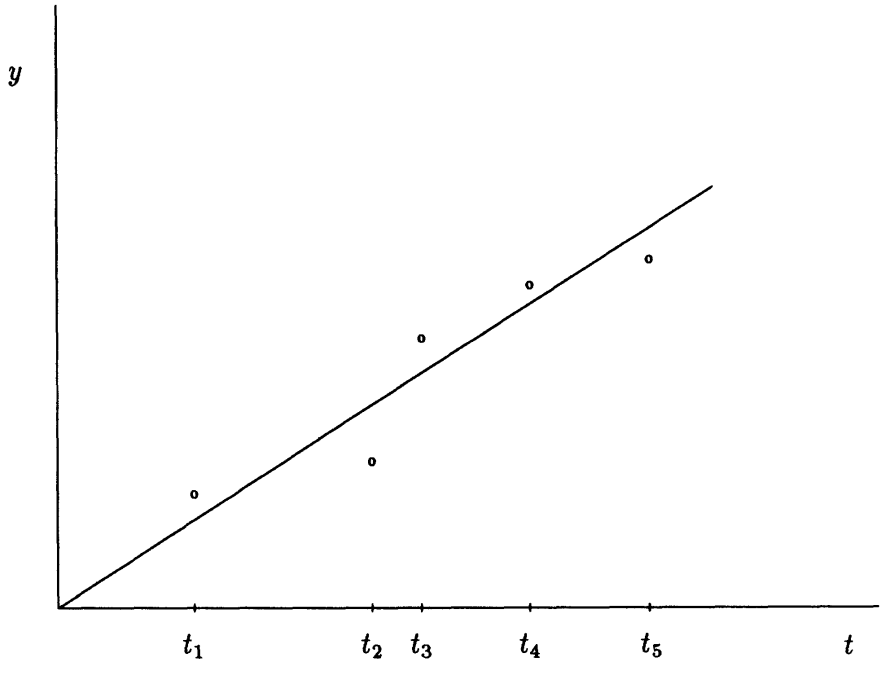


Abbildung 3.7: lineare Ausgleichsrechnung beim Ohm'schen Gesetz

Hier ist die Aufgabe, eine Gerade durch den Nullpunkt zu legen, die dem Verlauf der Messungen "möglichst nahe" kommt.

Gäbe es keine Meßfehler und würde das Modell exakt die Situation beschreiben, so sollten die Parameter $x_1, \ldots, x_n$ so zu bestimmen sein, daß

$$y_i = y(t_i) = \varphi(t_i; x_1, \ldots, x_n), \quad i = 1, \ldots, m$$

Tatsächlich sind die Messungen jedoch fehlerhaft und auch die Modellfunktionen stellen stets nur eine ungefähre Beschreibung der Wirklichkeit dar. (So gilt das Ohm'sche Gesetz annähernd nur in einem mittleren Temperaturbereich, was spätestens beim Durchglühen eines Drahtes deutlich wird.) Daher können wir nur fordern, daß

$$y_i \approx \varphi(t_i; x_1, \ldots, x_n), \quad i = 1, \ldots, m$$

Nun gibt es mehrere Möglichkeiten, die einzelnen Fehler

$$y_i - \varphi(t_i; x_1, \ldots, x_n), \quad i = 1, \ldots, m$$

zu gewichten. Bei der Gauß'schen Ausgleichsrechnung verwendet man die Euklidische Norm und versucht daher die Parameter $x_1, \ldots, x_n$ so zu bestimmen, daß

$$\sum_{i=1}^{m} (y_i - \varphi(t_i; x_1, \ldots, x_n))^2$$

minimal wird. Wir schreiben dafür auch einfach

$$\sum_{i=1}^{m} (y_i - \varphi(t_i; x_1, \ldots, x_n))^2 = \min \ . \tag{3.3.1}$$

In dieser Form werden die Fehler der einzelnen Messungen alle gleich gewichtet. Normalerweise sind die Messungen (t_i, y_i) jedoch unterschiedlich genau, sei es, weil die Meßapparatur in verschiedenen Bereichen unterschiedlich genau arbeitet, sei es, weil die Messungen einmal mit mehr und einmal mit weniger Sorgfalt durchgeführt wurden. Diese Toleranzen δy_i lassen sich in 3.3.1 einschließen, indem man die einzelnen Fehler mit der Toleranz wichtet, d.h.

$$\sum_{i=1}^{m} \left(\frac{y_i - \varphi(t_i; x_1, \ldots, x_n)}{\delta y_i} \right)^2 = \min \ .$$

Diese Form der Fehlerminimierung läßt auch eine vernünftige statistische Interpretation zu (sogenannte Σ-Modelle).

Wir betrachten hier nun den Spezialfall, daß die Modellfunktion φ linear in x ist, d.h.

$$\varphi(t; x_1, \ldots, x_n) = a_1(t)x_1 + \cdots + a_n(t)x_n \ ,$$

wobei $a_1, \ldots, a_n : \mathbf{R} \to \mathbf{R}$ beliebige Funktionen sind. In diesem Fall läßt sich das Problem (.) als *lineares Ausgleichsproblem* (LAP)

$$\|y - Ax\|_2 = \min$$

formulieren, wobei

$$\begin{aligned} y &:= (y_1, \ldots, y_m)^T \\ x &:= (x_1, \ldots, x_n)^T \\ A &:= (a_{ij}) \in Mat_{m,n}(\mathbf{R}) \ \text{ mit } \ a_{ij} := a_j(t_i) \end{aligned}$$

Als weitere Einschränkung betrachten wir hier nur den Fall

$$m \geq n$$

d.h. es liegen mehr Daten vor als Parameter zu bestimmen sind, was im statistischen Sinn vernünftig klingt. Wir erhalten somit die *Aufgabenstellung des linearen Ausgleichsproblems:*

Suche zu gegebenen $y \in \mathbf{R}^m$, $A \in Mat_{m,n}(\mathbf{R})$ *mit* $m \geq n$ *ein* $x \in \mathbf{R}^n$ *, so daß*

$$\|y - Ax\|_2 = \min \tag{3.3.2}$$

3.3.2 Lösung der Normalgleichungen

Die Existenz und Eindeutigkeit einer Lösung des LAP beschreibt der folgende Satz.

Satz 3.3.2 *(i)* x *ist genau dann Lösung des LAP 3.3.2, wenn es Lösung der* Normalgleichung

$$A^T Ax = A^T y$$

ist.

(ii) Das LAP ist genau dann eindeutig lösbar, wenn

$$\text{Rang } A = n$$

Beweis: (i) Wir betrachten die Funktion $f : \mathbf{R}^n \to \mathbf{R}^+$

$$f(x) := \|y - Ax\|_2^2 = < y - Ax, y - Ax >$$

Die Lösung des LAP besteht in der Bestimmung der Minima von f.

Falls $A \neq 0$, gilt $f(x) \to \infty$ für $\|x\| \to \infty$, so daß wir nur lokale Extrema zu betrachten brauchen. Diese sind gerade durch das Verschwinden des Gradienten ∇f von f charakterisiert

$$x \text{ lokales Extremum} \iff \nabla f(x) = (\frac{\partial}{\partial x_1}(x), \ldots, \frac{\partial}{\partial x_n}(x)) = 0$$

Nun gilt für zwei differenzierbare Funktionen $g, h : \mathbf{R}^n \to \mathbf{R}^m$ und das Funktional $f(x) := < g(x), h(x) >$, das

$$\nabla f(x) = g(x)^T Dh(x) + h(x)^T Dg(x) \; .$$

Speziell bei uns wegen $g(x) = h(x) = y - Ax$

$$\nabla f(x) = -2(y^T - x^T A^T)A$$

und daher

$$\begin{aligned} \nabla f(x) = 0 &\iff (y^T - x^T A^T)A = 0 \\ &\iff A^T Ax = A^T y \end{aligned}$$

Da die Hesse'sche Matrix von f

$$Hf(x) = 2A^TA$$

positiv semidefinit ist, folgt die Behauptung

$$x \text{ Minimum von } f \Longleftrightarrow A^TAx = A^Ty \ .$$

(ii) A^TA ist genau dann invertierbar, wenn Rang $A = n$. Die Normalgleichung besitzt daher genau dann eine eindeutige Lösung $x = (A^TA)^{-1}A^Ty$, wenn Rang $A = n$. □

Wir können also prinzipiell das LAP lösen, indem wir die Normalgleichung

$$A^TAx = A^Ty \tag{3.3.3}$$

lösen. Gehen wir davon aus, daß das LAP eindeutig lösbar ist, d.h. Rang $A = n$, so ist A^TA eine symmetrische positiv definite Matrix. Zur Lösung von 3.3.3 bietet sich daher das Cholesky-Verfahren an. Für den Aufwand zur Lösung des LAP mit Hilfe der Normalgleichung ergibt sich (Anzahl der Multiplikationen):

a) Berechnung von $A^TA :\sim \frac{1}{2}n^2m$

b) Cholesky-Zerlegung von $A^TA :\sim \frac{1}{6}n^3$

Für $m \gg n$ überwiegt der Anteil von a), so daß wir zusammen den Aufwand

$$\begin{aligned} &\sim \tfrac{1}{2}n^2m \quad \text{für} \quad m \gg n \text{ und} \\ &\sim \tfrac{2}{3}n^3 \quad \text{für} \quad m \approx n \end{aligned}$$

erhalten.

Wir haben jetzt das LAP gelöst, indem wir es zunächst in die Normalgleichung transformiert und dann mit den bewährten Methoden aus dem ersten Kapitel bearbeitet haben. Die im zweiten Kapitel entwickelte numerische Intuition läßt uns diesen Weg jedoch zweifelhaft erscheinen. In jedem zusätzlich eingeführten Schritt können Fehler entstehen, die dann bis zur Lösung propagiert werden. Daher ist es in den meisten Fällen besser, nach einer effizienten *direkten* Methode zu suchen. Als weiteren Kritikpunkt können wir einwenden, daß die Rundungsfehlerverstärkung in etwa durch die Kondition

$$\kappa_2(A^TA)$$

beschrieben wird. Für diese gilt:

Satz 3.3.3 *Für eine Matrix* $A \in Mat_{m,n}(\mathbf{R})$ *von maximalem Rang* $p = \min(m,n)$ *gilt*

$$\kappa_2(A^TA) = \kappa_2(A)^2$$

Beweis: Wir benutzten die in Kapitel 3.5 ausführlich beschriebene Singulärwertzerlegung von A

$$A = U\Sigma V^T \ , \ U \in O(m) \ , \ V \in O(n)$$

mit den Singulärwerten

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p > 0$$

Damit gilt

$$\kappa_2(A) = \sigma_1/\sigma_p$$

Andererseits erhalten wir die Singulärwertzerlegung von A^TA aus der von A durch

$$\begin{aligned} A^TA &= (U\Sigma V^T)^T U\Sigma V^T \\ &= V\Sigma U^T U\Sigma V^T \\ &= V\Sigma^2 V^T \end{aligned}$$

wobei $\Sigma^2 = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_p^2)$ und daher

$$\kappa_2(A^TA) = \frac{\sigma_1^2}{\sigma_p^2} = \kappa_2(A)^2$$

□

Die Matrix A^TA ist demnach sehr viel schlechter konditioniert als A. Hinzu kommt, daß die Matrizen, welche gewöhnlich bei LAP's auftreten, bereits schlecht konditioniert sind, so daß äußerste Vorsicht geboten ist und die weitere Verschlechterung durch den Übergang zu A^TA nicht hingenommen werden kann. Mildernd bleibt zu bemerken, daß bei entartetem Spaltenraum von A (Rangdefekt von A^TA) auch die rechte Seite ATy im entarteten Spaltenraum liegt – vgl. die Bemerkungen zur Lösung entarteter Gleichungssysteme in Kapitel 2.

3.3.3 Lösung mittels QR-Zerlegung

Als Alternative zur Lösung der Normalgleichungen bietet sich die direkte Behandlung von A und y mit Orthogonaltransformationen an. Tatsächlich drängt die Invarianz der Euklidischen Norm bezüglich orthogonaler Transformationen die Anwendung der QR-Zerlegung zur Lösung des linearen Ausgleichsproblems geradezu auf. Wir gehen aus von der QR-Zerlegung

$$\underbrace{Q_n \cdot \cdots \cdot Q_1}_{Q^T} A = \begin{bmatrix} * & \cdots & * \\ & \ddots & \vdots \\ & & * \\ & 0 & \end{bmatrix} = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

der Matrix $A \in Mat_{m,n}(\mathbf{R})$, $m \geq n$, mit Hilfe der Householder-Reflexionen

$$Q_j \in O(m)$$

Der einzige Unterschied zur QR-Zerlegung, wie wir sie bisher kennengelernt haben besteht darin, daß wir die Householder-Transformationen Q_j auf die rechteckige Matrix A anwenden und gegenüber der QR-Zerlegung quadratischer Matrizen eine weitere Householder-Reflexion benötigen, um die letzte Spalte in die gewünschte Form zu bringen.

Beispiel 3.3.4 $n = 3$, $m = 5$

$$\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \xrightarrow{Q_1} \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \\ 0 & x & x \\ 0 & x & x \end{bmatrix} \xrightarrow{Q_2} \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & x \\ 0 & 0 & x \end{bmatrix} \xrightarrow{Q_3} \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Die Lösung des LAP erhalten wir damit sehr einfach:

Satz 3.3.5 *Sei* $A \in Mat_{m,n}(\mathbf{R})$, $m \geq n$, *von maximalem Rang,* Rang $(A) = m$, $y \in \mathbf{R}^m$, *und* $Q \in O(m)$ *mit*

$$Q^T A = \begin{bmatrix} R \\ 0 \end{bmatrix} , \; Q^T y = \begin{pmatrix} c \\ d \end{pmatrix} , \; c \in \mathbf{R}^n , \; d \in \mathbf{R}^{m-n}$$

Dann ist

$$x^* := R^{-1} c$$

die Lösung des LAP

$$\|y - Ax\|_2 = \min .$$

Beweis: Da $Q \in O(m)$, gilt für alle $x \in \mathbf{R}^m$, daß

$$\begin{aligned} \|y - Ax\|_2^2 &= \|Q^T(y - Ax)\|_2^2 \\ &= \|\begin{pmatrix} c - Rx \\ d \end{pmatrix}\|_2^2 \\ &= \|c - Rx\|_2^2 + \|d\|_2^2 \geq \|d\|_2^2 \end{aligned}$$

Der erste Summand $\|c - Rx\|_2^2$ verschwindet genau für $x = R^{-1}c$. □

Für den Aufwand erhalten wir bei dieser Methode (wieder Anzahl der Multiplikationen)

$$\begin{aligned} &\sim n^2 m \quad \text{für} \quad m \gg n \\ &\sim \tfrac{2}{3} n^3 \quad \text{für} \quad m \approx n \end{aligned}$$

Für $m \approx n$ benötigen wir also in etwa den gleichen Aufwand wie bei dem Normalgleichungsansatz mit Cholesky-Zerlegung. Für $m \gg n$ schneidet die QR–Zerlegung um einen Faktor 2 schlechter ab.

Bemerkung 3.3.6 Als weiteren Vorteil der QR-Zerlegung zur Lösung des LAP weisen wir auf die Möglichkeit des *Rangentscheides* mit Hilfe der Subkondition $sc(A)$ hin, die bei der QR-Zerlegung leicht berechnet werden kann:

$$\text{eps} \cdot sc(A) = \text{eps} \frac{|r_{11}|}{|r_{nn}|} \geq 1 \implies \text{(numerischer) Rang } (A) < n$$

3.4 QR-Algorithmus für symmetrische Eigenwertprobleme

Nachdem wir in Kapitel 2.2 herausgefunden haben, daß das Eigenwertproblem für symmetrische Matrizen gutkonditioniert ist ($\kappa_{abs} = 1$), interessiert uns natürlich, wie dieses Problem effektiv gelöst werden kann. In diesem Abschnitt sei $A \in \mathrm{Mat}_n(\mathbf{R})$ eine reelle symmetrische Matrix, $A^T = A$. Wir wissen, daß A in diesem Fall nur reelle Eigenwerte $\lambda_1, \ldots, \lambda_n \in \mathbf{R}$ besitzt und eine Orthonormalbasis $\eta_1, \ldots, \eta_n \in \mathbf{R}^n$ aus Eigenvektoren $A\eta_i = \lambda_i \eta_i$ existiert, d.h.

$$Q^T A Q = \Lambda := \mathrm{diag}\,(\lambda_1, \ldots, \lambda_n) \quad \text{mit} \quad Q := [\eta_1, \ldots, \eta_n] \in O(n) \tag{3.4.1}$$

Die erste Idee, die einem in den Sinn kommen könnte, wäre, Q direkt in endlich vielen Schritten zu bestimmen. Da die Eigenwerte die Wurzeln des charakteristischen Polynoms sind, hätte man damit auch ein endliches Verfahren zur Bestimmung der Nullstellen von Polynomen beliebigen Grades (im Fall symmetrischer Matrizen nur mit reellen Wurzeln) gefunden. Dem steht der *Satz von Abel* entgegen. Ein solches Verfahren (basierend auf den Operationen $+, -, \cdot, /$ und Wurzelziehen) gibt es nicht.

Die zweite Idee, die von 3.4.1 nahegelegt wird, ist, A durch Konjugation (z. B. mit orthogonalen Matrizen) der Diagonalgestalt näherzubringen, da die Eigenwerte invariant unter Ähnlichkeitstransformationen sind. Versucht man, eine symmetrische Matrix A durch Konjugation mit Householder-Matrizen auf Diagonalgestalt zu bringen, so erweist sich dies schnell als unmöglich.

$$\begin{bmatrix} x & \ldots & x \\ \vdots & & \vdots \\ \vdots & & \vdots \\ x & \ldots & x \end{bmatrix} \xrightarrow{Q_1 \cdot} \begin{bmatrix} x & x & \ldots & x \\ 0 & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & x & \ldots & x \end{bmatrix} \xrightarrow{\cdot Q_1^T} \begin{bmatrix} x & 0 & \ldots & 0 \\ \vdots & x & \ldots & x \\ \vdots & \vdots & & \vdots \\ x & x & \ldots & x \end{bmatrix}$$

Was die Multiplikation mit der Householder-Transformation von links geschafft hat, wird bei der Multiplikation von rechts wieder zerstört. Anders sieht es aus, wenn wir A nur auf *Tridiagonalgestalt* bringen wollen

$$\begin{bmatrix} x & \ldots & x \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \vdots & & \vdots \\ x & \ldots & x \end{bmatrix} \xrightarrow{P_1 \cdot} \begin{bmatrix} x & x & \ldots & x \\ x & \vdots & & \vdots \\ 0 & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & x & \ldots & x \end{bmatrix} \xrightarrow{\cdot P_1^T} \begin{bmatrix} x & x & 0 & \ldots & 0 \\ x & x & \ldots & \ldots & x \\ 0 & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ 0 & x & \ldots & \ldots & x \end{bmatrix} \tag{3.4.2}$$

Hier stören sich die Householder-Transformationen von links und rechts nicht.

Lemma 3.4.1 *Sei $A \in \mathrm{Mat}_n(\mathbf{R})$ symmetrisch. Dann existiert eine orthogonale Matrix Q (welche das Produkt von $n-2$ Householder-Reflexionen ist), so daß*

$$QAQ^T$$

Tridiagonalgestalt hat.

Beweis: Wir iterieren den in 3.4.2 gezeigten Prozeß und erhalten so Householder-Reflexionen $P_1, \ldots, P_{n-2}$, so daß

$$\underbrace{P_{n-2} \cdots P_1}_{=Q} A \underbrace{P_1 \cdots P_{n-2}}_{=Q^T} = \begin{bmatrix} x & x & & \\ x & \ddots & \ddots & \\ & \ddots & \ddots & x \\ & & x & x \end{bmatrix}$$

□

Damit haben wir unser Ausgangsproblem transformiert auf die Frage nach den Eigenwerten einer symmetrischen Tridiagonalmatrix.

Die Idee des folgenden Algorithmus geht auf Rutishauser zurück. Er hat ausprobiert, was passiert, wenn man die Faktoren der LR-Zerlegung einer Matrix $A = LR$ vertauscht, $A' = RL$, und diesen Prozeß von Zerlegen und Vertauschen iteriert. Dabei zeigte sich, daß in vielen Fällen die Folge der so konstruierten Matrizen gegen die Diagonalmatrix Λ der Eigenwerte konvergiert.

Beim QR-Algorithmus verwendet man anstatt einer LR-Zerlegung eine QR-Zerlegung. Diese existiert immer (keine Permutation nötig) und ist vor allen Dingen inhärent stabil (s. (3.1)).

Algorithmus 3.4.2 *QR-Algorithmus (Francis 1959, Kublanovskaja 1961)*
Definiere eine Folge $\{A_k\}$ von Matrizen durch

$$\begin{array}{lllll} a) & A_1 & = & A & \\ b) & A_k & = & Q_k R_k, & QR\text{-Zerlegung} \\ c) & A_{k+1} & := & R_k Q_k & \end{array}$$

Bemerkung 3.4.3

1) *Die Matrizen A_s sind konjugiert, $A_s \sim A$, mit einer orthogonalen Ähnlichkeitstransformation.*

2) *Ist A symmetrisch, so auch alle A_s.*

3) *Ist A symmetrisch und tridiagonal, so auch alle A_s.*

Beweis: zu 1) Sei $A = QR$ und $A' = RQ$. Dann gilt $QA'Q^T = QRQQ^T = QR = A$.
zu 2) Die Transformationen der Form

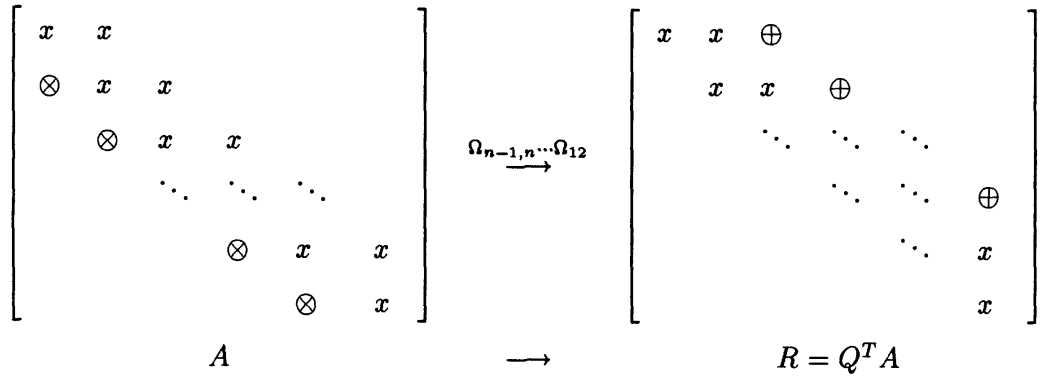
$$A \to B^T A B, \quad B \in GL(n)$$

sind die Basiswechsel für Bilinearformen und erhalten somit die Symmetrie. Insbesondere gilt dies für orthogonale Ähnlichkeitstransformationen, wie auch direkt aus

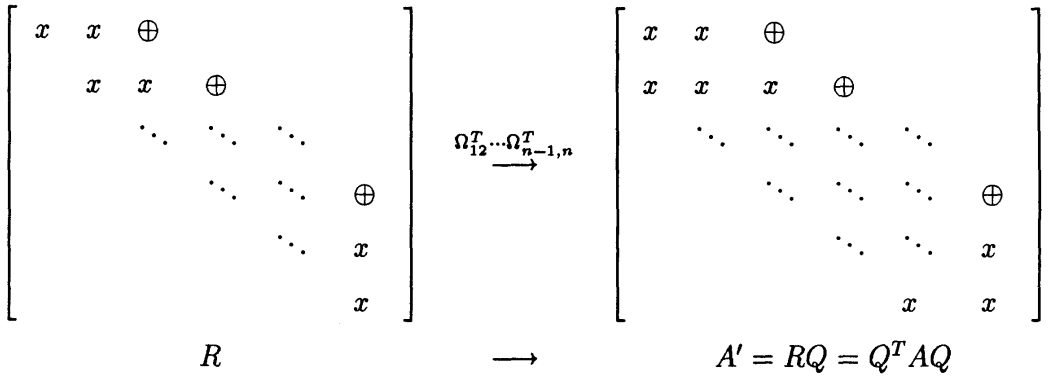
$$(A')^T = (A')^T Q^T Q = Q^T R^T Q^T Q = Q^T A^T Q = Q^T A Q = A'$$

folgt.

zu 3) Sei A symmetrisch und tridiagonal. Wir realisieren Q mit $n-1$ Givens-Rotationen $\Omega_{12},\ldots,\Omega_{n-1,n}$, so daß $Q^T = \Omega_{n-1,n}\cdots\Omega_{12}$ ($\otimes$ zu eliminieren, $\oplus$ neu erzeugtes (fill in-) Element)



$A \longrightarrow R = Q^T A$



$R \longrightarrow A' = RQ = Q^T AQ$

Nach 2) wissen wir, daß A' wieder symmetrisch sein muß und daher alle $\oplus$-Einträge verschwinden. Also ist A' wieder tridiagonal. □

Die Konvergenzeigenschaft zeigen wir nur für den einfachen Fall, daß die Beträge der Eigenwerte von A paarweise verschieden sind.

Satz 3.4.4 *Sei* $A \in \mathrm{Mat}_n(\mathbf{R})$ *symmetrisch mit den Eigenwerten* $\lambda_1,\ldots,\lambda_n$, *so daß*

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n| > 0 \tag{3.4.3}$$

und A_k, Q_k, R_k *wie in (3.4) definiert. Dann gilt mit* $A_k = (a_{ij}^{(k)})$

$$\begin{aligned} a) \quad & \lim_{k\to\infty} Q_k = I \\ b) \quad & \lim_{k\to\infty} R_k = \Lambda \\ c) \quad & a_{i,j}^{(k)} = O(|\tfrac{\lambda_i}{\lambda_j}|^k) \ \textit{für}\ i,j = 1,\ldots,n \end{aligned}$$

Beweis: Der hier gegebene Beweis geht auf Wilkinson zurück. Wir zeigen zunächst, daß

$$A^k = \underbrace{Q_1\cdots Q_k}_{=:P_k}\underbrace{R_k\cdots R_1}_{=:U_k} \quad \text{für } k = 1,2,\ldots$$

Für $k = 1$ ist die Behauptung klar, da $A = A_1 = Q_1 R_1$. Andererseits folgt aus der Konstruktion der A_k, daß

$$\begin{aligned} A_{k+1} &= Q_{k+1} R_{k+1} = Q_k^T \cdots Q_1^T A Q_1 \cdots Q_k \\ &= P_k^{-1} A P_k \end{aligned}$$

und daher der Induktionsschritt

$$\begin{aligned} A^{k+1} &= A A^k = A P_k U_k \\ &= P_k Q_{k+1} R_{k+1} U_k = P_{k+1} U_{k+1} \end{aligned}$$

Da $P_k \in O(n)$ orthogonal und U_k obere Dreiecksmatrix, können wir die QR-Zerlegung $A^k = P_k U_k$ von A^k durch die QR-Zerlegung der $A_1, \ldots, A_k$ ausdrücken. Ferner gilt

$$A^k = Q \Lambda^k Q^T, \quad \Lambda^k = \operatorname{diag}(\lambda_1^k, \ldots, \lambda_n^k)$$

Wir nehmen nun der Einfachheit halber an, Q hätte eine LR-Zerlegung,

$$Q = LU$$

wobei L unipotente untere und U obere Dreiecksmatrix ist. Dies können wir immer erreichen, indem wir A mit einer geeigneten Permutation konjugieren. Damit gilt

$$\begin{aligned} A^k &= Q \Lambda^k L U \\ &= Q(\Lambda^k L \Lambda^{-k})(\Lambda^k U) \end{aligned} \tag{3.4.4}$$

Für die unipotente untere Dreiecksmatrix $\Lambda^k L \Lambda^{-k}$ gilt

$$(\Lambda^k L \Lambda^{-k})_{ij} = \ell_{ij} \left(\frac{\lambda_i}{\lambda_j} \right)^k$$

Insbesondere verschwinden alle Nicht-Diagonalelemente für $k \to \infty$, d.h.

$$\Lambda^k L \Lambda^{-k} = I + E_k \text{ mit } E_k \to 0 \text{ für } k \to \infty$$

Eingesetzt in 3.4.4 folgt

$$A^k = Q(I + E_k)\Lambda^k U$$

Nun wenden wir auch auf $I + E_k$ (formal) ein QR-Zerlegung

$$I + E_k = \tilde{Q}_k \tilde{R}_k$$

an, wobei alle Diagonalelemente von $\tilde{R}_k$ positiv seien. Dann folgt aus der Eindeutigkeit der QR-Zerlegung und $\lim_{k \to \infty} E_k = 0$, daß

$$\tilde{Q}_k, \tilde{R}_k \to I \text{ für } k \to \infty$$

Damit haben wir eine zweite QR-Zerlegung von A^k hergeleitet, da

$$A^k = (Q\tilde{Q}_k)(\tilde{R}_k \Lambda^k U)$$

Bis auf die Vorzeichen in der Diagonale gilt daher

$$P_k = Q_k \tilde{Q}_k, \quad U_k = \tilde{R}_k \Lambda^k U$$

und es folgt

$$\begin{aligned} Q_k &= P_{k-1}^T P_k = \tilde{Q}_{k-1}^T Q^T Q \tilde{Q}_k \\ &= \tilde{Q}_{k-1}^T \tilde{Q}_k \to I \text{ für } k \to \infty \\ R_k &= U_k U_{k-1}^{-1} = \tilde{R}_k \Lambda^k U U^{-1} \Lambda^{-(k-1)} \tilde{R}_{k-1}^{-1} \\ &= \tilde{R}_k \Lambda \tilde{R}_{k-1}^{-1} \to \Lambda \text{ für } k \to \infty \end{aligned}$$

und

$$\lim_{k\to\infty} A_k = \lim_{k\to\infty} Q_k R_k = \lim_{k\to\infty} R_k \ .$$

□

Bemerkung 3.4.5

1) *Eine genauere Analyse zeigt, daß das Verfahren auch für mehrfache Eigenwerte* $\lambda_i = \cdots = \lambda_j$ *konvergiert.*

2) *Falls* $\lambda_i = -\lambda_{i+1}$ *konvergiert das Verfahren nicht. Es bleiben Blöcke stehen.*

Liegen zwei Eigenwerte λ_i, λ_{i+1} betragsmäßig dicht beieinander, so konvergiert das Verfahren nur sehr mäßig. Dies kann mit Hilfe der sogenannten *Shift-Strategien* verbessert werden. Im Prinzip versucht man, die beiden Eigenwerte dichter an den Nullpunkt zu schieben und so den Quotienten $|\lambda_{i+1}/\lambda_i|$ zu verkleinern. Dazu verwendet man für jeden Iterationsschritt k einen *Shift-Parameter* σ_k und definiert die Folge $\{A_k\}$ durch

$$\begin{aligned} &\text{a)} \quad A_1 &&= A \\ &\text{b)} \quad A_k - \sigma_k I &&= Q_k R_k, \quad QR\text{-Zerlegung} \\ &\text{c)} \quad A_{k+1} &&= R_k Q_k + \sigma_k I \end{aligned}$$

Es folgt wie oben

1) $A_{k+1} = Q_k^T A_k Q_k \sim A_k$

2) $(A - \sigma_1 I) \cdots (A - \sigma_k I) = Q_1 \cdots Q_k R_k \cdots R_1$

Die Folge $\{A_k\}$ konvergiert gegen Λ mit der Geschwindigkeit

$$a_{i,j}^{(k)} = O\left(\left| \frac{\lambda_i - \sigma_1}{\lambda_j - \sigma_1} \right| \cdots \left| \frac{\lambda_i - \sigma_{k-1}}{\lambda_j - \sigma_{k-1}} \right| \right)$$

Die σ_k sollten möglichst nahe an den Eigenwerten λ_i, λ_{i+1} liegen, um die Konvergenzbeschleunigung zu erreichen.

Wilkinson hat folgende Shift-Strategie vorgeschlagen: Wir gehen aus von einer symmetrischen Tridiagonalmatrix A. Falls dann das untere Ende der Tridiagonalmatrix A_k von der Form

$$\begin{array}{ccc} \ddots & \ddots & \\ \ddots & d_{n-1}^{(k)} & e_n^{(k)} \\ & e_n^{(k)} & d_n^{(k)} \end{array}$$

ist, so hat die 2×2-Eckmatriz zwei Eigenwerte, von denen wir denjenigen als σ_k wählen, der näher an $d_n^{(k)}$ liegt.

Besser als diese *expliziten* Shiftstrategien sind insbesondere für schlecht skalierte Matrizen die *impliziten Shift-Verfahren*, für die wir wieder auf [10] bzw. [3] verweisen.

Neben den Eigenwerten interessieren wir uns auch für die Eigenvektoren, die sich wie folgt berechnen lassen:

Algorithmus 3.4.6 *Bestimmung sämtlicher Eigenwerte und Eigenvektoren einer symmetrischen Matrix*

a) Reduziere das Problem auf Tridiagonalgestalt

$$A \to A_1 = QAQ^T, \quad A_1 \text{ symmetrich und tridiagonal}$$

b) Approximiere die Eigenwerte mit dem QR-Algorithmus mit Givens-Rotationen angewandt auf A_1

$$\Omega A_1, \Omega^T \doteq \Lambda, \quad \Omega \text{ Produkt aller Givens-Rotationen } \Omega_{ij}^{(k)}$$

Nun gilt

$$\begin{aligned} A &= Q^T A_1 Q \doteq Q^T \Omega^T \Lambda \Omega Q \\ &\doteq (\Omega Q)^T \Lambda (\Omega Q) \end{aligned}$$

und der Vergleich mit (3.4.1) liefert:

c) Die Eigenvektoren sind die Spalten von ΩQ

$$\Omega Q = [\eta_1, \ldots, \eta_n]$$

Der Aufwand beträgt

für a)	:	$\frac{2}{3}n^3$	Multiplikationen
für b)	:	$O(n^2)$	Multiplikationen

3.5 Singulärwertzerlegung

Ein sehr nützliches Mittel zur Analyse von Matrizen stellt die sogenannte *Singulärwertzerlegung* einer Matrix $A \in Mat_{m,n}(\mathbf{R})$ dar. Wir zeigen zunächst die Existenz einer solchen Zerlegung und listen einige Eigenschaften auf. Anschließend werden wir uns ansehen, wie sich die Singulärwerte günstig berechnen lassen.

3.5.1 Existenz und Eigenschaften der Singulärwertzerlegung

Satz 3.5.1 *Sei* $A \in Mat_{m,n}(\mathbf{R})$. *Dann gibt es orthogonale Matrizen* $U \in O(m)$ *und* $V \in O(n)$, *so daß*

$$U^T A V = \sum = diag(\sigma_1, \ldots, \sigma_p) \in Mat_{m,n}(\mathbf{R}) ,$$

wobei $p = \min(m,n)$ *und* $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$.

Beweis: Es genügt zu zeigen, daß es $U \in O(m)$ und $V \in O(n)$ gibt, so daß

$$U^T A V = \begin{bmatrix} \sigma & 0 \\ 0 & B \end{bmatrix}$$

Die Behauptung folgt dann durch Induktion. Sei $\sigma := \|A\|_2$. Dann gibt es $v \in \mathbf{R}^n$, $u \in \mathbf{R}^n$, so daß

$$Av = \sigma u \quad , \quad \|u\|_2 = \|v\|_2 = 1$$

Erweitere $\{v\}$ zu einer Orthonormalbasis von $\{v = v_1, \ldots, v_n\}$ des $\mathbf{R}^n$ und $\{u\}$ zu einer Orthonormalbasis $\{u = u_1, \ldots, u_m\}$ des $\mathbf{R}^m$. Dann sind

$$V := (v_1, \ldots, v_n) \ , \ U := (u_1, \ldots, u_m)$$

orthogonale Matrizen, $V \in O(n)$, $U \in O(m)$ und $U^T A V$ von der Form

$$U^T A V = \begin{bmatrix} \sigma & w^T \\ 0 & B \end{bmatrix} =: A_1 \quad \text{mit } w \in \mathbf{R}^{n-1}$$

Da

$$\|A_1 \begin{pmatrix} \sigma \\ w \end{pmatrix}\|_2^2 \geq (\sigma^2 + w^T w)^2 \quad \text{und} \quad \|\begin{pmatrix} \sigma \\ w \end{pmatrix}\|_2^2 = \sigma^2 + w^T w$$

gilt

$$\sigma^2 = \|A\|_2^2 = \|A_1\|_2^2 \geq \sigma^2 + w^T w$$

und daher $w = 0$, also

$$U^T A V = \begin{bmatrix} \sigma & 0 \\ 0 & B \end{bmatrix}$$

□

Definition 3.5.2 *Die Zerlegung* $U^T A V = \sum$ *heißt* Singulärwertzerlegung *von* A, *die* σ_i *sind die* Singulärwerte *von* A.

Korollar 3.5.3 *Mit* $p = \min(m,n)$ *gilt:*

1) $AV_i = \sigma_i U_i$, $A^T U_i = \sigma_i V_i$ *für* $i = 1, \ldots, p$

2) $\|A\|_F^2 = \sigma_1^2 + \ldots + \sigma_p^2$

3) $\|A\|_2 = \sigma_1$

4) Falls $\sigma_1 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_p = 0$, *so gilt*

(i) Rang $A = r$

(ii) $\ker A =$ span $\{V_{r+1}, \ldots, V_n\}$

(iii) im $A =$ span $\{U_1, \ldots, U_r\}$

5) $\kappa_2(A) = \sigma_1/\sigma_p$

6) Die $\sigma_1^2, \ldots, \sigma_p^2$ *sind die von Null verschiedenen Eigenwerte von* A^TA *und* AA^T *zu den Eigenvektoren* $V_1, \ldots, V_p$ *bzw.* $U_1, \ldots, U_p$.

Beweis: Übung, siehe Aufgabe 3.3. □

3.5.2 Berechnung der Singulärwerte einer Matrix

Nach Korollar 3.5.3 sind die Singulärwerte σ_i einer Matrix $A \in \mathrm{Mat}_n(\mathbf{R})$ die Wurzeln der Eigenwerte von A^TA:

$$\sigma_i(A) = \sqrt{\lambda_i(A^TA)} \tag{3.5.1}$$

Da A^TA eine symmetrische positiv semidefinite Matrix ist, ist das Eigenwertproblem von A^TA und damit auch die Bestimmung der Singulärwerte von A gutkonditioniert. Auch bietet sich mit (3.5.1) eine Berechnungmethode für die $\sigma_i(A)$ an. Wie bei der linearen Ausgleichsrechnung ist dieser (Um-) Weg jedoch ungeeignet; was sich an folgendem Beispiel leicht nachvollziehen läßt:

Beispiel 3.5.4 *Rechne auf vier Ziffern gerundet*

$$A = A^T = \begin{pmatrix} 1.005 & 0.995 \\ 0.995 & 1.005 \end{pmatrix}, \quad \sigma_1 = \lambda_1 = 2, \quad \sigma_2 = \lambda_2 = 0.01$$

Über den Zugang mit A^TA *erhalten wir*

$$\mathrm{fl}(A^TA) = \begin{pmatrix} 2.000 & 2.000 \\ 2.000 & 2.000 \end{pmatrix}, \quad \tilde{\sigma}_1^2 = 4., \quad \tilde{\sigma}_2^2 = 0.$$

Wir suchen daher ein Verfahren, welches nur auf der Matrix A operiert.

Lemma 3.5.5 *Sei* $A \in \mathrm{Mat}_{m,n}(\mathbf{R})$, *und* $P \in O(m)$, $Q \in O(n)$ *orthogonale Matrizen. Dann haben* A *und*

$$B := PAQ$$

die gleichen Singulärwerte.

Beweis: Einfache Übung. □

Im Fall der Singulärwertberechnung dürfen wir also die Matrix A von links und rechts mit beliebigen orthogonalen Matrizen multiplizieren. Damit ist es leicht möglich, A auf *Bidiagonalgestalt* zu bringen.

Lemma 3.5.6 *Für jede Matrix* $A \in \mathrm{Mat}_{m,n}(\mathbf{R})$ *existieren orthogonale Matrizen* $P \in O(m)$, $Q \in O(n)$, *so daß*

$$PAQ = \left[\begin{array}{cccc} x & x & & \\ & \ddots & \ddots & \\ & & \ddots & x \\ & & & x \\ \hline 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{array}\right] = \begin{bmatrix} B \\ 0 \end{bmatrix}$$

wobei B *eine (quadratische) Bidiagonalmatrix ist.*

Beweis: Wir veranschaulichen die Konstruktion von P und Q mit Householder-Matrizen:

$$\begin{bmatrix} x & \cdots & \cdots & x \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ x & \cdots & \cdots & x \end{bmatrix} \xrightarrow{P_1 \cdot} \begin{bmatrix} x & x & \cdots & x \\ 0 & & & \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & x & \cdots & x \end{bmatrix} \xrightarrow{\cdot Q_1} \begin{bmatrix} x & x & 0 & \cdots & 0 \\ 0 & x & & & x \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ 0 & x & & \cdots & x \end{bmatrix}$$

$$\xrightarrow{P_2 \cdot} \begin{bmatrix} x & x & 0 & \cdots & 0 \\ 0 & x & x & \cdots & x \\ & 0 & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & x & \cdots & x \end{bmatrix} \longrightarrow \cdots \longrightarrow \begin{bmatrix} x & x & & & \\ & x & x & & \\ & & \ddots & \ddots & \\ & & & \ddots & x \\ & & & & x \end{bmatrix}$$

Damit gilt dann

$$\begin{pmatrix} B \\ 0 \end{pmatrix} = \underbrace{P_n \cdot \cdots \cdot P_1}_{=:P} A \underbrace{Q_1 \cdot \cdots \cdot Q_{n-2}}_{=:Q} .$$

□

Ist B bidiagonal, so ist B^TB tridiagonal. Daher untersucht man den QR-Algorithmus für die Tridiagonalmatrix B^TB und versucht, eine vereinfachte Version zu finden, die ausschließlich auf B operiert

$$\begin{aligned} A_1 &:= B_1^TB_1, \quad B_1 := B \\ A_2 &= \Omega_{12}B^TB\Omega_{12}^T = \underbrace{(B_1\Omega_{12}^T)^T}_{B_2^T}\underbrace{B_1\Omega_{12}^T}_{B_2} \end{aligned}$$

Damit erhält man

$$B_2 = B_1\Omega_{12}^T = \begin{bmatrix} x & x & & & & \\ \oplus & x & x & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & x & \\ & & & & x & \end{bmatrix}$$

Spielt man den QR-Algorithmus für B^TB auf diese Weise auf B zurück, so zeigt sich, daß das Verfahren folgendem Eliminationsprozeß ("chasing") entspricht.

$$\begin{bmatrix} x & x & z_3 & & & & \\ z_2 & x & x & z_5 & & & \\ & z_4 & x & x & z_7 & & \\ & & \ddots & \ddots & \ddots & \ddots & \\ & & & z_{2n-6} & x & x & z_{2n-3} \\ & & & & z_{2n-4} & x & x \\ & & & & & z_{2n-2} & x \end{bmatrix} \tag{3.5.2}$$

eliminiere	z_2	(Givens von links)	$\rightarrow$	erzeugt	z_3
eliminiere	z_3	(Givens von rechts)	$\rightarrow$	erzeugt	z_4
$\vdots$					
eliminiere	z_{2n-3}	(Givens von rechts)	$\rightarrow$	erzeugt	z_{2n-2}
eliminiere	z_{2n-2}	(Givens von links)			

Man "jagt" also dem im ersten Schritt erzeugten Element z_2 entlang der beiden Diagonalen nach und beseitigt dabei mit Givens-Rotationen abwechselnd von links und von rechts die neu entstehenden Komponenten. Zum Schluß hat die Matrix wieder Bidiagonalgestalt und wir haben einen Iterationsschritt des QR-Verfahrens für B^TB nur auf B ausgeführt. Nach Satz 3.4.1 gilt

$$B_k^TB_k \rightarrow \Lambda = \operatorname{diag}(\sigma_1^2, \ldots, \sigma_n^2) = \sigma^2 \quad \text{für} \quad k \rightarrow \infty$$

Daher konvergiert die Folge B_k gegen die Diagonalmatrix der Singulärwerte von B

$$B_k \to \sigma \text{ für } k \to \infty$$

Zusammengefaßt erhalten wir:

Algorithmus 3.5.7 *Bestimmung der Singulärwerte von* A

1) *Bringe* $A \in \mathrm{Mat}_{m,n}(\mathbf{R})$ *mit Hilfe orthogonaler Transformationen (z.B. Householder-Reflexionen) von links und rechts auf Bidiagonalgestalt.*

$$PAQ = \begin{pmatrix} B \\ 0 \end{pmatrix}, \quad P \in O(m), \quad Q \in O(n), \quad B \in \mathrm{Mat}_n(\mathbf{R}) \text{ obere Bidiagonalmatrix}$$

2) *Führe den* QR*-Algorithmus für* B^TB *nach dem "chasing"-Muster (3.4) auf* B *aus und erhalte so eine Folge von Bidiagonalmatrizen* $\{B_k\}$*, die gegen die Diagonalmatrix* σ *der Singulärwerte konvergiert.*

Für den Aufwand zählen wir

für 1)	$\sim \frac{4}{3}n^3$	Multiplikationen (für $m = n$)
für 2)	$O(n^2)$	Multiplikationen

3.6 Übungen

Aufgabe 3.1 Bestimmen Sie die Eigenwerte, Eigenvektoren und die Determinante einer Householder-Transformation

$$Q = I - 2\frac{vv^T}{v^T v}.$$

Aufgabe 3.2 Zeigen Sie, daß für $u, v \in \mathbf{R}$ gilt:

$$(I + uv^T)^{-1} = I - \frac{uv^T}{1 + v^T u}, \quad \text{falls } u^T v \neq -1$$
$$I + uv^T \text{ singulär, falls } u^T v \neq -1$$

Aufgabe 3.3 Zeigen Sie die Eigenschaften der die Singulärwertzerlegung aus Korollar 3.5.3.

Aufgabe 3.4 In der Chemie werden häufig sogenannte Reaktionsgeschwindigkeitskonstanten K_i $(i = 1, \ldots, m)$ bei Meßtemperaturen T_i mit einer absoluten Meßgenauigkeit (Toleranz) δK_i gemessen. Mit Hilfe des Arrhenius-Gesetzes

$$K_i = A \cdot \exp\left(-\frac{E}{RT_i}\right)$$

bestimmt man daraus im Sinne der kleinsten Fehlerquadrate die beiden Parameter, den präexponentieller Faktor A und die Aktivierungsenergie E, wobei die allgemeine Gaskonstante R vorgegeben ist. Formulieren Sie das gestellte nichtlineare Problem als *lineares* Ausgleichsproblem. Welche Vereinfachungen erhält man für die beiden Spezialfälle

a) $\delta K_i = \epsilon K_i$ (konstanter relativer Fehler)

b) $\delta K_i = \delta K$ (konstanter absoluter Fehler)

Aufgabe 3.5 Programmieren Sie das Orthogonalisierungsverfahren nach Householder ohne Spaltentausch für (m, n)-Matrizen, $m \geq n$, und lösen Sie damit das lineare Anfangswertproblem aus der 3. Aufgabe für folgende zwei Datensätze:

4 Lösung nichtlinearer Gleichungen

Bis jetzt haben wir uns fast aussschließlich mit linearen Problemen beschäftigt, in diesem Kapitel wollen wir uns nichtlinearen Gleichungen

$$f(x) = 0$$

zuwenden. Dabei sollten wir uns genau vor Augen führen, was wir unter dem "Lösen" einer solchen Gleichung verstehen wollen. So kennt jeder aus der Schule für die quadratische Gleichung

$$f(x) := x^2 - 2px + q = 0$$

die analytische geschlossen darstellbare Lösung

$$x_{1,2} = p \pm \sqrt{p^2 - q}$$

Tatsächlich haben wir dadurch das Problem der Lösung einer quadratischen Gleichung nur transformiert in die Berechnung einer Wurzel, mit anderen Worten, die Lösung einer quadratischen Gleichung der Form

$$f(x) := x^2 - c = 0$$

(wobei hier $c = |p^2 - q|$). In jedem Fall müssen wir noch ein Verfahren angeben, wie eine Lösung von (4.0.1) bzw. (4.0.2) zu bestimmen ist. Vom numerischen Standpunkt ist eine Lösung daher eher eine Tätigkeit. Bei der Lösung nichtlinearer Gleichungen beschränken wir uns hier auf zwei Methoden, die *Fixpunktiteration* und das *Newton-Verfahren.*

4.1 Allgemeine Fixpunktiteration

Die Idee der Fixpunktiteration besteht darin, die Gleichung

$$f(x) = 0$$

äquivalent in eine Fixpunktgleichung

$$\phi(x) = x$$

umzuformen und mit der Iterationsvorschrift

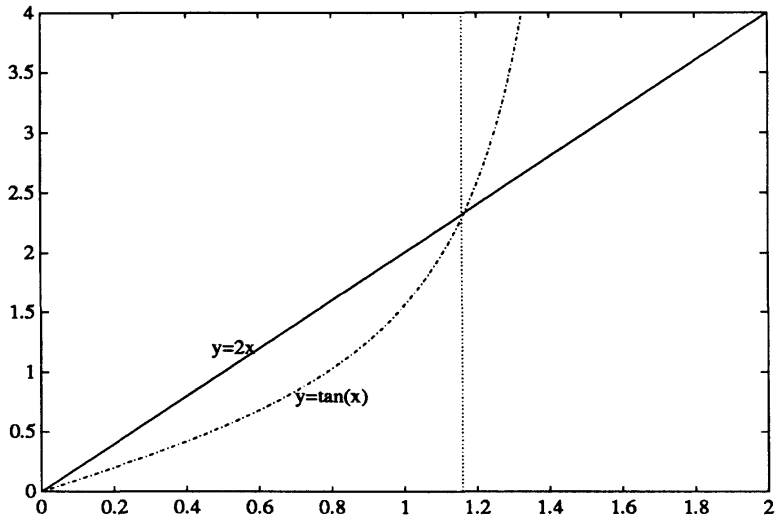
$$x_{k+1} = \phi(x_k), \quad k = 0, 1, \ldots$$

für einen gegebenen Startwert x_0 eine Folge $\{x_k\}$ zu konstruieren in der Hoffnung, daß diese gegen eine Lösung (beider Gleichungen) konvergiert, d.h.

$$\lim_{k \to \infty} x_k = x^* \text{ mit } \phi(x^*) = x^*$$

Beispiel 4.1.1 Wir betrachten die Gleichung

$$f(x) := 2x - \tan\ x = 0$$



Versuchen wir die Gleichung graphisch zu lösen
so erhalten wir im Intervall $[0.5, 2]$ die ungefähre Lösung

$$x^* \approx 1.2$$

Die Gleichung (.) können wir leicht auf zweifache Weise in eine Fixpunktgleichung umformen:

$$\text{a)} \quad f(x) = 0 \quad \Leftrightarrow \quad x = \tfrac{1}{2}\tan\, x =: \phi_1(x)$$

$$\text{b)} \quad f(x) = 0 \quad \Leftrightarrow \quad x = \arctan(2x) =: \phi_2(x)$$

Probieren wir die beiden zugehörigen Fixpunktiterationen mit dem Startwert $x_0 = 1.2$ aus, so ergeben sich folgende Zahlen

k	$x_{k+1} = \frac{1}{2}\tan\, x_k = \phi_1(x_k)$	$x_{k+1} = \arctan(2x_k) = \phi_2(x_k)$
0	1.2	1.2
1	1.2860	1.1760
2	$1.70\ldots > \pi/2$	1.1687
3		1.1665
4		1.1658
5		1.1656
6		1.1655
7		1.1655

Wie man sieht, divergiert die erste Folge ($\tan\, x$ hat einen Pol bei $\pi/2$ und $x_2 > \pi/2$), die zweite dagegen konvergiert, wobei ungefähr bei jedem zweiten Iterationsschritt eine gültige Ziffer mehr auftritt.

Wie ist dieses Verhalten zu erklären? Dazu betrachten wir allgemein Folgen $\{x_k\}$, die durch eine *Iterationsfunktion* ϕ

$$x_{k+1} = \phi(x_k)$$

gegeben sind. Wollen wir die Differenz zweier Folgenglieder

$$|x_{k+1} - x_k| = |\phi(x_k) - \phi(x_{k-1})|$$

durch die Differenz $|x_k - x_{k-1}|$ der vorhergehenden abschätzen, (wobei wir natürlich sofort an die geometrische Reihe denken), so führt dies zu folgender Definition

Definition 4.1.2 *Sei $I = [a,b] \subset \mathbf{R}$ ein Intervall und $\phi : I \to \mathbf{R}$. ϕ ist* kontrahierend auf I, *falls es ein $\Theta > 0$ gibt, so daß*

a) $|\phi(x) - \phi(y)| \leq \Theta|x - y|$ *für alle* $x, y \in I$

b) $\Theta < 1$

Die Bedingung a) nennt man dabei auch eine *Lipschitz-Bedingung* für ϕ. Eine kontrahierende Abbildung erfüllt also die Lipschitz-Bedingung a) mit einer *Lipschitz-Konstanten* $\Theta < 1$. Für stetig differenzierbare ϕ ist diese Bedingung leicht zu überprüfen:

Lemma 4.1.3 *Ist $\phi : I \to \mathbf{R}$ stetig differenzierbar, $\phi \in C^1(I)$, so gilt*

$$\sup_{x,y \in I} \frac{|\phi(x) - \phi(y)|}{|x - y|} = \sup_{z \in I} |\phi'(z)| < \infty$$

Beweis: Dies ist eine einfache Anwendung des Mittelwertsatzes: Für alle $x, y \in I$, $x < y$, existiert ein $\xi \in [x,y]$, so daß

$$\phi(x) - \phi(y) = \phi'(\xi)(x - y)$$

□

Satz 4.1.4 *(Spezialfall des Banach'schen Fixpunktsatzes). Sei $I = [a,b] \subset \mathbf{R}$ ein Intervall und*

$$\phi : I \to I$$

eine kontrahierende Abbildung mit Lipschitz-Konstante $\Theta < 1$. Dann folgt:

a) Es existiert genau ein Fixpunkt x^* von ϕ, $\phi(x^*) = x^*$

b) Für jeden Startwert $x_0 \in I$ konvergiert die Fixpunktiteration

$$x_{k+1} = \phi(x_k), \quad k = 0, 1, \ldots$$

gegen x^* mit der Abschätzung:

$$|x^* - x_k| \leq \frac{\Theta^k}{1 - \Theta}|x_1 - x_0| \, .$$

Beweis: zu b) Es gilt für $x_0 \in I$, daß

$$|x_{k+1} - x_k| = |\phi(x_k) - \phi(x_{k-1})| \leq \Theta |x_k - x_{k-1}|$$

und daher induktiv

$$|x_{k+1} - x_k| \leq \Theta^k |x_1 - x_0|$$

Wir wollen zeigen, daß $\{x_k\}$ eine Cauchy-Folge ist und betrachten daher

$$\begin{aligned} |x_{k+m} - x_k| &\leq |x_{k+m} - x_{k+m-1}| + \cdots + |x_{k+1} - x_k| \\ &\leq \underbrace{(\Theta^{k+m-1} + \Theta^{k+m-2} + \cdots + \Theta^k)}_{=\Theta^k(1+\Theta+\cdots+\Theta^{m-1})} |x_1 - x_0| \\ &\leq \frac{\Theta^k}{1-\Theta} |x_1 - x_0|, \end{aligned}$$

wobei wir die Dreiecksungleichung und die Abschätzung für die geometrische Reihe $\sum_{k=0}^{\infty} \Theta^k = 1/(1-\Theta)$ benutzt haben. Also ist

$$\lim_{k,\ell \to \infty} |x_k - x_\ell| = 0$$

und daher $\{x_k\}$ eine Cauchy-Folge, die in dem vollständigen Raum der reellen Zahlen gegen

$$x^* := \lim_{k \to \infty} x_k$$

konvergiert. x^* ist aber auch Fixpunkt von ϕ, da

$$\begin{aligned} |x^* - \phi(x^*)| &= |x^* - x_{k+1} + x_{k+1} - \phi(x^*)| \\ &= |x^* - x_{k+1} + \phi(x_k) - \phi(x^*)| \\ &\leq |x^* - x_{k+1}| + |\phi(x_k) - \phi(x^*)| \\ &\leq |x^* - x_{k+1}| + \Theta |x^* - x_k| \quad \to 0 \text{ für } k \to \infty \end{aligned}$$

zu a) Sind x^*, y^* zwei Fixpunkte, so gilt

$$0 \leq |x^* - y^*| = |\phi(x^*) - \phi(x^*)| \leq \Theta |x^* - y^*|$$

Da $\Theta < 1$ ist dies nur für $|x^* - y^*| = 0$ möglich. Daher ist der Fixpunkt von ϕ eindeutig bestimmt. □

Bemerkung 4.1.5 *In dem Beweis wurden nur die Dreiecksungleichung und die Vollständigkeit als Eigenschaften von* $\mathbf{R}$ *benutzt. Er gilt daher allgemein in metrischen Räumen* (X, d) *für kontrahierende Abbildungen*

$$\phi : Y \subset X \to Y$$

$$d(\phi(x), \phi(y)) \leq \Theta d(x, y) \text{ für } x, y \in Y$$

mit $\Theta < 1$.

Wie wir oben an dem einfachen Beispiel $f(x) = 2x - \tan x$ gesehen haben, muß man zur Lösung des nichtlinearen Problems $f(x) = 0$ eine geeignete Fixpunktiteration unter mehreren möglichen auswählen. Dies ist im allgemeinen nicht einfach zu bewerkstelligen. Zur Beurteilung der *Konvergenzgeschwindigkeit* definieren wir den Begriff der *Konvergenzordnung* einer Folge $\{x_k\}$.

Definition 4.1.6 *Eine Folge* $\{x_k\}, x_k \in \mathbf{R}^n$, *konvergiert mit der* Ordnung $p \geq 1$ *gegen* x^*, *falls es eine Konstante* $C \geq 0$ *gibt, so daß*

$$\begin{aligned} &a) \quad \|x_{k+1} - x^*\| \leq C\|x_k - x^*\|^p \\ &b) \quad C < 1, \ \textit{falls } p = 1 \end{aligned}$$

Im Fall $p = 1$ *sprechen wir auch von* linearer, *für* $p = 2$ *von* quadratischer Konvergenz. *Weiter heißt* $\{x_k\}$ superlinear *konvergent, falls*

$$\begin{aligned} &a) \quad \|x_{k+1} - x^*\| \leq C_k\|x_k - x^*\|, \ \textit{wobei} \\ &b) \quad C_k \geq 0 \ \textit{mit} \ \lim_{k\to\infty} C_k = 0 \ . \end{aligned}$$

Bemerkung 4.1.7 *Häufig definiert man die Konvergenzordnung* p *aus Gründen einer einfacheren Analysis alternativ durch die analogen Ungleichungen für die Iterierten*

$$\|x_{k+1} - x_k\| \leq C\|x_k - x_{k-1}\|^p$$

bzw.

$$\|x_{k+1} - x_k\| \leq C_k\|x_k - x_{k-1}\|$$

für die superlineare Konvergenz.

Die Ordnung eines Iterationsverfahrens definieren wir als die kleinste Ordnung der Iterationsfolgen $\{x_k\}$ für alle Startwerte x_0 aus einer vorgegebenen Menge. So konvergiert wegen

$$|x_{k+1} - x^*| = |\phi(x_k) - \phi(x^*)| \leq \Theta|x_k - x^*|$$

und $\Theta < 1$ die Fixpunktiteration

$$x_{k+1} = \phi(x_k)$$

für eine kontrahierende Abbildung ϕ linear. Von einem guten Iterationsverfahren erwarten wir jedoch, daß es zumindest *superlinear* oder *linear mit einer kleinen Konstante* $C \ll 1$ konvergiert. Wie wir sehen werden, konvergiert das nun folgende Newton-Verfahren sogar quadratisch.

4.2 Gewöhnliches Newton-Verfahren für skalare Gleichungen

Im eindimensionalen Fall beruht das Newton-Verfahren zur Lösung einer nichtlinearen Gleichung

$$f(x) = 0$$

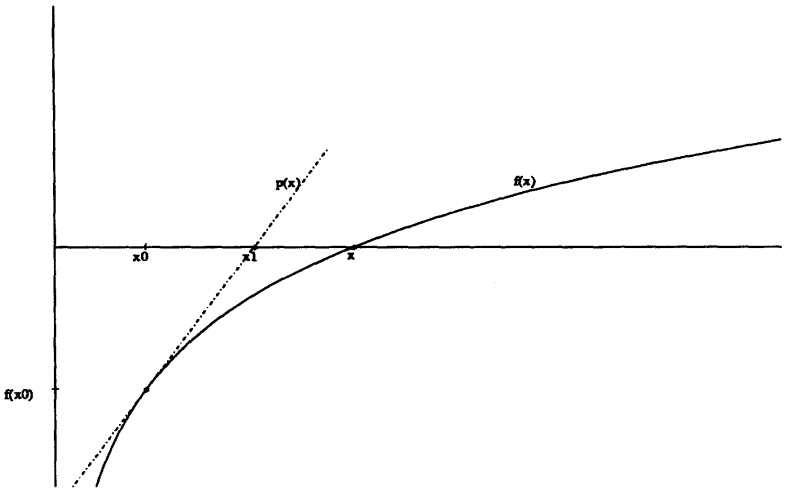


Abbildung 4.8: Idee des Newton-Verfahrens

auf folgender Idee. Wir approximieren die Funktion f durch ihre Tangente $p(x)$ im Startpunkt x_0 und berechnen anstelle des Schnittpunktes des Graphen von f mit der x-Achse den der Tangente $p(x)$ mit der x-Achse

Das Tangentenpolynom ist gerade

$$p(x) = f(x_0) + f'(x_0)(x - x_0)$$

(d.h. nichts anderes als die ersten beiden Terme der Taylor-Entwicklung von f um x_0) und die Nullstelle x_1 von $p(x)$ läßt sich im Fall von $f'(x_0) \neq 0$ durch

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

berechnen. Das *Newton-Verfahren* besteht nun in der wiederholten Anwendung dieser Vorschrift, d.h.

$$x_{k+1} := x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots$$

für einen gegebenen Startwert x_0. Offensichtlich ist dies eine spezielle Fixpunktiteration mit der Iterationsfunktion

$$\phi(x) := x - \frac{f(x)}{f'(x)}$$

die wir natürlich nur bilden können, falls f differenzierbar ist und $f'(x) \neq 0$ (zumindest in einer Umgebung der Lösung). Der Einfachheit halber setzen wir im folgenden Satz sogar dreimalige stetige Differenzierbarkeit voraus.

Satz 4.2.1 *Sei f dreimal stetig differenzierbar, $f \in C^3$, x^* eine Lösung von f, $f(x^*) = 0$, und*

$$f'(x^*) \neq 0$$

Dann existiert eine Umgebung $I \subset \mathbf{R}$ von x^*, so daß für jeden Startwert $x_0 \in I$ die Newton-Folge $\{x_k\}$ mit

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots$$

gegen x^* quadratisch konvergiert, d.h.

$$|x_{k+1} - x^*| \leq C|x_k - x^*|^2$$

Beweis: Wir wenden den Fixpunktsatz und Lemma 4.1.3 auf die Iterationsfunktion ϕ an. Es gilt:

$$\phi'(x) = 1 - \frac{f'(x)f'(x) - f(x)f''(x)}{f'(x)^2} = \frac{f''(x)f(x)}{f'(x)^2}$$

Da $\phi(x^*) = x^*, \phi'(x^*) = 0$ und ϕ' stetig existiert eine Umgebung I von x^*, so daß

$$\begin{array}{ll} \text{a)} & f'(x) \neq 0 \text{ für } x \in I \\ \text{b)} & \phi(x) \in I \text{ für } x \in I \\ \text{c)} & \Theta := \sup\limits_{x \in I} |\phi'(x)| < 1 \end{array}$$

Nach Lemma ?? ist ϕ daher kontrahierend und aufgrund des Fixpunktsatzes konvergiert $\{x_k\}$ gegen x^*. Da $\phi'(x^*) = 0$ ist $\{x_k\}$ zumindest superlinear konvergent. Die quadratische Konvergenz sieht man wie folgt. Da $f \in C^3$, ist $\phi \in C^2$ und daher für alle $x \in I$

$$\begin{aligned} \phi(x) &= \phi(x^*) + \phi'(x^*)(x - x^*) + \frac{1}{2}\phi''(\xi)(x - x^*)^2 \\ &= \phi(x^*) + \frac{1}{2}\phi''(\xi)(x - x^*)^2 \end{aligned}$$

für ein (von x abhängendes) ξ zwischen x und x^*. Daraus folgt

$$\begin{aligned} |x_{k+1} - x^*| &= |\phi(x_k) - \phi(x^*)| \\ &= |\frac{1}{2}\phi''(\xi)(x_k x^*)^2| \\ &\leq \underbrace{\frac{1}{2}\sup_{x \in I} |\phi''(x)|}_{=:C} \cdot |x_k - x^*|^2 \end{aligned}$$

und somit die quadratische Konvergenz. □

Für "hinreichend nahe" Startwerte x_0 konvergiert also das Newton-Verfahren unter der Voraussetzung $f'(x^*) \neq 0$. Man bezeichnet dies auch als *lokale Konvergenz.* Obiger Satz läßt sich daher auch kurz und knapp durch den Merksatz

"Das Newton-Verfahren konvergiert lokal quadratisch"

formulieren.

Beispiel 4.2.2 Als Anwendung kehren wir zurück zur Berechnung der Quadratwurzel, d.h. der Lösung der Gleichung

$$f(x) := x^2 - c = 0$$

Im Rechner werde die Zahl c als Gleitkommazahl

$$c = a2^p \ , \quad 0.5 < a \leq 1$$

durch die Mantisse a und den Exponenten p dargestellt. Damit gilt

$$\sqrt{c} = \begin{cases} \sqrt{a} \cdot 2^m & , \quad \text{falls } p = 2m \\ (\sqrt{a}\sqrt{0.5}) \cdot 2^m & , \quad \text{falls } p = 2m+1, \end{cases}$$

wobei

$$\sqrt{0.5} < \sqrt{a} \leq 1 \ .$$

Wird $\sqrt{0.5} = 1/\sqrt{2} \approx 0.71$ ein für allemal gesondert berechnet und abgespeichert, so bleibt nur noch das Problem

$$f(x) := x^2 - a = 0 \ \text{ für } \ a \in (\frac{1}{2}, 1]$$

zu lösen. Da

$$f'(x) = 2x \neq 0 \ \text{ für } \ c \in (\frac{1}{2}, 1]$$

ist das Newton-Verfahren anwendbar und wir erhalten als Iterationsfunktion

$$\begin{aligned} \phi(x) &= x - \frac{f(x)}{f'(x)} = x - \frac{x^2 - a}{2x} \\ &= x - \frac{x}{2} + \frac{a}{2x} \\ &= \frac{1}{2}\left(x + \frac{a}{x}\right) \end{aligned}$$

also die Newton-Iteration

$$x_{k+1} := \frac{1}{2}\left(x_k + \frac{a}{x_k}\right)$$

Die Division durch 2 kann dabei sogar sehr billig durch die Subtraktion von 1 im Exponenten realisiert werden, so daß nur eine Division pro Iterationsschritt durchgeführt werden muß.

Beispiel 4.2.3 $a = 0.81, \quad x_0 = 1$

k	x_k
0	1.0000000000
1	0.9050000000
2	0.9000138122
3	0.90000000001

Die quadratische Konvergenz ist in etwa gleichbedeutend mit einer Verdopplung der Anzahl der gültigen Ziffern.

4.3 Newton-Verfahren für Systeme nichtlinearer Gleichungen

Das Newton-Verfahren läßt leicht auf Systeme nichtlinearer Gleichungen der Form

$$F(x) = 0, \quad F : \mathbf{R}^n \to \mathbf{R}^n$$

übertragen, wobei F eine stetig differenzierbare Funktion mit gewissen Zusatzeigenschaften ist. Häufig liegt die Jacobi-Matrix

$$F'(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x) & \dots & \frac{\partial f_1}{\partial x_n}(x) \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(x) & \dots & \frac{\partial f_n}{\partial x_n}(x) \end{bmatrix} \in Mat_n(\mathbf{R})$$

nicht geschlossen vor, sondern wird durch Differenzenquotienten

$$\frac{\partial f_i}{\partial x_j}(x) \to \frac{f_i(x_1, \dots, x_j + \delta, \dots, x_n) - f_i(x_1, \dots, x_n)}{\delta}$$

approximiert. Man beachte dabei die problematische Wahl von δ: Ist δ zu klein, tritt Auslöschung auf; wählt man δ zu groß, ist die Approximation der Ableitung schlecht. Als günstiger Wert hat sich

$$\delta = O(\sqrt{\text{eps}})$$

erwiesen.

Die Taylor-Entwicklung von F um einen Startwert x^0 ergibt für eine Lösung x^*

$$0 = F(x^*) = F(x^0) + F'(x^0)(x^* - x^0) + \quad \text{Terme höherer Ordnung}$$

also

$$x^* - x^0 \doteq -F'(x^0)^{-1} F(x^0),$$

falls der Startwert x^0 "nahe bei" x^* liegt und $F'(x^0)$ nicht singulär ist. Dies inspiriert die sogenannte *gewöhnliche Newton-Iteration* $(k = 0, 1, \dots)$

$$\begin{aligned} &\text{a)} \quad F'(x^k)\Delta x^k = -F(x^k) \\ &\text{b)} \quad x^{k+1} = x^k + \Delta x^k \end{aligned}$$

Bei dieser Form wird deutlich, daß nicht etwa $F'(x^k)^{-1}$ in jedem Iterationsschritt berechnet werden muß, sondern nur die Lösung Δx^k des linearen Gleichungssystems a). Wir haben also die Lösung eines nichtlinearen Gleichungssystems zurückgeführt auf eine Folge linearer Gleichungssysteme.

Auch das mehrdimensionale Newton-Verfahren konvergiert lokal quadratisch. Aus der Fülle möglicher Konvergenzsätze greifen wir hier nur einen heraus, dessen Voraussetzungen sich dadurch auszeichnen, daß sie invariant unter linearen Transformationen sind, d.h. falls man die Funktion F durch

$$F \to G := AF$$

ersetzt, wobei $A \in GL(n)$ eine nicht singuläre Matrix ist.

Satz 4.3.1 *Sei $D \subset \mathbf{R}^n$ offen und konvex und $F : D \to \mathbf{R}^n$ eine stetig differenzierbare Funktion. Weiter gebe es Konstanten $w \geq 0$ und $\alpha \geq 0$, so daß*

a) $\|F'(y)^{-1}(F'(x + t(y - x)) - F'(x))(x - y)\| \leq tw\|x - y\|^2$ *für alle* $x, y \in D$ *und* $t \in [0, 1]$

b) $\|F'(x^0)^{-1}F(x^0)\| \leq \alpha$ *für einen Startwert* $x^0 \in D$

c) $h := \dfrac{\alpha w}{2} < 1$

d) $\bar{B}_\rho(x^0) := \{x \in \mathbf{R}^n | \|x - x^0\| \leq \rho\} \subset D$, *wobei* $\rho := \dfrac{\alpha}{1 - h}$

Dann ist die Newton-Iteration

$$\begin{aligned} F'(x^k)\Delta x^k &= -F(x^k) \\ x^{k+1} &= x^k + \Delta x^k \end{aligned}$$

mit Startwert x^0 wohldefiniert und konvergiert gegen eine Lösung $x^* \in \bar{B}_\rho(x^0)$ von F.

Bemerkung 4.3.2 Die Bedingung a) nennt man auch eine *affin invariante Lipschitz-Bedingung.*

Beweis: Wir wollen wieder zeigen, daß $\{x^k\}$ eine Cauchy-Folge ist. Dazu zeigen wir zunächst, daß

$$\|x^{k+1} - x^k\| \leq \frac{1}{2}w\|x^k - x^{k-1}\|^2$$

Eine einfache Umformung ergibt

$$\begin{aligned} \|x^{k+1} - x^k\| &= F'(x^k)^{-1}F(x^k)\| \\ &= \|F'(x^k)^{-1}(F(x^k) - F(x^{k-1}) - F'(x^{k-1})\Delta x^{k-1})\| \end{aligned}$$

da $F(x^{k-1}) = -F'(x^{k-1})\Delta x^{k-1}$. Wir benutzen nun die Tatsache, daß für eine differenzierbare Funktion $\varphi : \mathbf{R} \to \mathbf{R}^n$ gilt

$$\varphi(1) - \varphi(0) = \int_0^1 \varphi'(t)dt \ .$$

Setzen wir speziell

$$\varphi(t) := F(x^{k-1} + t\Delta x^{k-1})$$

so gilt

$$\varphi(1) - \varphi(0) = F(x^k) - F(x^{k-1}) \ \text{ und } \ \varphi'(t) = F'(x^{k-1} + t\Delta x^{k-1})\Delta x^{k-1}$$

und daher

$$\begin{aligned} \|x^{k+1} - x^k\| &= \|F'(x^k)^{-1}\int_0^1 (F'(x^{k-1} + t\Delta x^{k-1}) - F'(x^{k-1}))\Delta x^{k-1}dt\| \\ &= \|\int_0^1 F'(x^k)^{-1}(F'(x^{k-1} + t\Delta x^{k-1}) - F'(x^{k-1}))\Delta x^{k-1}dt\| \\ &\leq \int_0^1 \|F'(x^k)^{-1}(F'(x^{k-1} + t\Delta x^{k-1}) - F'(x^{k-1}))\Delta x^{k-1}\|dt \\ &\leq \int_0^1 wt\|\Delta x^{k-1}\|^2 dt \\ &= \frac{1}{2}w\|\Delta x^{k-1}\|^2 \end{aligned}$$

Induktiv folgt damit, daß

$$\|\Delta x^k\| = \|x^{k+1} - x^k\| \leq \alpha h^{2^k - 1}$$

Für $k = 0$ ist dies gerade Voraussetzung. Der Induktionsschritt folgt aus

$$\begin{aligned}
\|\Delta x^k\| &\leq \frac{1}{2} w \|\Delta x^{k-1}\|^2 \\
&\leq \frac{1}{2} w (\alpha h^{2^{k-1}-1})^2 \\
&= \frac{1}{2} w \alpha^2 h^{2^k - 2} \qquad \text{(Induktionsannahme)} \\
&= \alpha (\frac{1}{2} w \alpha) h^{2^k - 2} \\
&= \alpha h^{2^k - 1} .
\end{aligned}$$

Mit Hilfe der geometrischen Reihe (wegen c) gilt $h < 1$) folgt daraus sofort

$$\text{(i)} \qquad \|x^m - x^n\| \leq \frac{\alpha h^{2^n - 1}}{1 - h^{2^n}} \quad \text{für} \quad m \geq n$$

$$\text{(ii)} \qquad \|x^n - x^0\| \leq \frac{\alpha}{1 - h} = \rho$$

Wegen (ii) ist die Newton-Folge wohldefiniert in $\bar{B}_\rho(x^0)$ und wegen (i) eine Cauchy-Folge mit Limes

$$x^* := \lim_{k \to \infty} x^k \in \bar{B}_\rho(x^0)$$

x^* ist auch Lösung von F, denn

$$\|F(x^k)\| = \|F'(x^k)\Delta x^k\| \leq \|F'(x^k)\| \underbrace{\|\Delta x^k\|}_{\to 0} \to 0 \text{ für } k \to \infty$$

da F' als stetig und daher beschränkt auf dem Kompaktum $\bar{B}_\rho(x^0)$ angenommen wurde. □

Wir haben bisher nur die lokale Konvergenz des Newton-Verfahren gezeigt. Es bleibt die Frage, wie eine Lösung x^* von F gefunden werden kann, wenn kein hinreichend guter Startwert x^0, nahe bei x^*, zu haben ist. In dieser Situation eröffnen sich im wesentlichen zwei Möglichkeiten:

Globale Newton-Verfahren

Die erste Möglichkeit besteht darin, zu den sogenannten *gedämpften* oder *globalen* Newton-Verfahren überzugehen. Dabei ersetzt man die gewöhnliche Newton-Iteration durch

$$\text{a)} \qquad F'(x^k)\Delta x^k = -F(x^k)$$

$$\text{b)} \qquad x^{k+1} = x^k + \lambda_k \Delta x^k, \quad 0 < \lambda_k < 1,$$

wobei die λ_k geeignet zu wählende *Dämpfungsfaktoren* sind. Wer Näheres dazu wissen möchte, dem sei das Buch [5] empfohlen.

Fortsetzungsmethoden

Die Idee ist hier, sich von einem bereits gelösten Problem

$$G(x) = 0$$

Zug um Zug zur Lösung des eigentlichen Problems

$$F(x) = 0$$

durchzuarbeiten. Dazu konstruiert man eine Problemschar

$$F(x, \tau) = 0$$

mit einem Parameter $\tau \in \mathbf{R}$, so daß

$$F(x,0) = G(x), \quad F(x,1) = F(x)$$

Die einfachste Möglichkeit ist dabei vom "identischen Problem" $G(x) = x$ auszugehen und

$$F(x,\tau) := \tau F(x) + (1-\tau)x, \quad \tau \in [0,1]$$

zu betrachten. Diese lineare Einbettung des ursprünglichen Problems in eine Problemschar führt allerdings in den meisten Fällen nicht zum Ziel. Vielmehr müssen für problemangepaßt sinnvollere Einbettungen gesucht werden.

Sei nun x_0^* eine bekannte Lösung zu $\tau_0 = 0$. Dann versucht man weitere Lösungen x_ν^* zu den Parameterwerten

$$\tau_1 < \tau_2 < \ldots < \tau_\nu < \ldots$$

zu finden, indem man

$$x_\nu^0 := x_{\nu-1}^*$$

als Startwert für das Newton-Verfahren für die Funktion $F(x, \tau_\nu)$ ansetzt. Konvergiert das Verfahren, so geht man zum nächsten Parameterwert (mit gleicher oder vergrßerter Schrittweite über, im anderen Fall reduziert man den Abstand zwischen den Parameterwerten und versucht es zum Beispiel nochmals mit

$$\tau_\nu' := \tau_{\nu-1} + \frac{\tau_\nu - \tau_{\nu-1}}{2}$$

Im eindimensionalen Fall läßt sich diese Technik wie folgt veranschaulichen.

Diese Methode bezeichnet man auch als *einfache Fortsetzungsmethode*, was ausgefeiltere Techniken vermuten läßt. Dem ist auch so, wie man z.B. in [5] nachlesen kann.

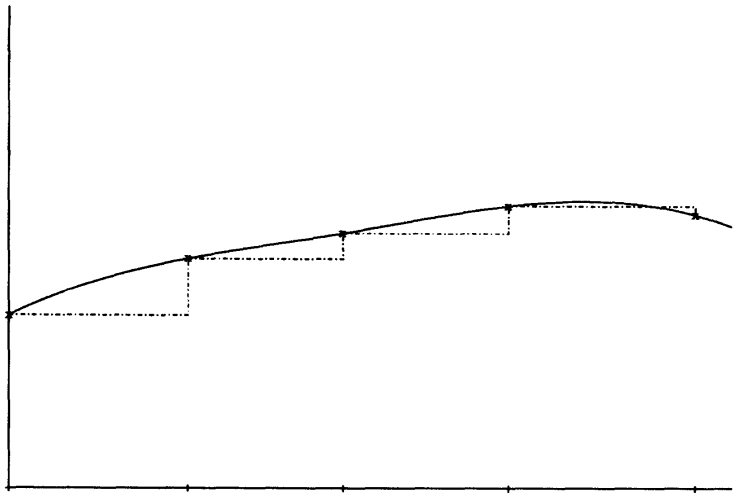


Abbildung 4.9: Prinzip der einfachen Fortsetzungmethode

4.4 Übungen

Aufgabe 4.1 Erklären Sie das unterschiedliche Konvergenzverhalten der beiden in der Vorlesung beschriebenen Fixpunktiterationen zur Lösung von

$$f(x) = 2x - \tan x = 0.$$

Analysieren Sie dabei auch die Konvergenzgeschwindigkeit des zweiten Verfahrens.

Aufgabe 4.2 Zur Bestimmung des Fixpunktes x^* der stetig differenzierbaren Abbildung ϕ mit $|\phi'(x^*)| \neq 1$ seien die folgenden Iterationsvorschriften für $k = 0, 1, \ldots$ definiert:

(I) $x_{k+1} := \phi(x_k)$

(II) $x_{k+1} := \phi^{-1}(x_k)$

Zeigen Sie, daß mindestens eine der beiden Iterationen lokal konvergiert.

Aufgabe 4.3 Eine Funktion $f \in C^1[a, b]$ habe die einfache Nullstelle $x^* \in [a, b]$. Durch die drei Stützpunkte

$$(a, f_a), \quad (c, f_c), \quad (b, f_b), \quad \text{mit} \quad a < c < b, \quad f_a f_b < 0$$

ist ein quadratisches Interpolationspolynom $p(x)$ eindeutig bestimmt.

a) Zeigen Sie, daß p in $[a, b]$ genau ein einfache Nullstelle y besitzt.

b) Konstruieren Sie ausgehend von einer formalen Prozedur

$$y = y(a, b, c, f_a, f_b, f_c),$$

welche die Nullstelle von p in $[a, b]$ in Abhängigkeit von den Stützpunkten berechnet, einen Algorithmus zur Bestimmung von x^* auf eine vorgegebene Genauigkeit eps.

Aufgabe 4.4 Zur Konvergenzbeschleunigung eines linear konvergenten Fixpunktverfahren im $\mathbf{R}^1$

$$x_{i+1} := \phi(x_i), \quad x_0 \text{ vorgegeben}, \; x^* \text{ Fixpunkt}$$

kann man die sogenannte Δ^2-Methode von Aitken verwenden. Dabei wird zu der Folge $\{x_i\}$ die transformierte Folge $\{\bar{x}_i\}$

$$\bar{x}_i := x_i - \frac{(\Delta x_i)^2}{\Delta^2 x_i}$$

berechnet, wobei Δ der Differenzenoperator $\Delta x_i := x_{i+1} - x_i$ ist.

a) Zeigen Sie: Gilt für die Folge $\{x_i\}$ und $x_i \neq x^*$, daß

$$x_{i+1} - x^* = (k + \delta_i)(x_i - x^*)$$

wobei $|k| \neq 1$ und $\{\delta_i\}$ eine Nullfolge, $\lim_{i\to\infty} \delta_i = 0$, so existiert die Folge $\{\bar{x}_i\}$ für hinreichend große i und hat die Eigenschaft

$$\lim_{i\to\infty} \frac{\bar{x}_i - x^*}{x_i - x^*} = 0$$

b) Bei der Realisierung des Verfahrens berechnet man nur x_0, x_1, x_2 und $\bar{x}_0$ und startet dann die Fixpunktiteration mit $\bar{x}_0$ als verbessertem Startwert (Methode von Steffensen). Probieren Sie diese Fixpunktiteration an unseren bewährten Beispielen

$$\phi_1(x) := (\tan x)/2 \quad \text{und} \quad \phi_1(x) := \arctan 2x$$

mit dem Startwert $x_0 = 1.2$ aus.

5 Polynom–Interpolation und Approximation

Häufig tritt in der numerischen Mathematik die Situation auf, daß statt einer Funktion $f : \mathbf{R} \to \mathbf{R}$ nur einige diskrete Funktionswerte

$$f_i = f(t_i), \quad i = 0, \ldots, n$$

an bestimmten Punkten gegeben sind. Dies ist zum Beispiel der Fall, wenn die Funktion f aus experimentellen Daten bestimmt wird. Auch bei den meisten Verfahren zur Lösung von Differentialgleichungen wird eine gesuchte Funktion $f(t)$ nur an endlich vielen Stellen $t_0, \ldots, t_n$ berechnet.

Ist man an dem gesamten Verlauf der Funktion interessiert, so sollte man aus den gegebenen Punkten eine Funktion φ konstruieren, die sich möglichst wenig von der ursprünglichen Funktion f unterscheidet, also

$$\varphi - f \text{ "klein"}$$

Bemerkung 5.0.1 Ursprünglich trat das Problem bei der Berechnung von Funktionswerten, z.B. $\sin, \ell n, \ldots$ mit Hilfe von Tabellen auf. Heute spielt die Interpolation eine Rolle bei den sogenannten *Extrapolationsmethoden*, die wir am Beispiel der Romberg-Quadratur im Kapitel 6.2 kennenlernen werden. Der größte Anwendungsbereich ist inzwischen die Computer-Graphik unter den Schlagkürzeln CAD (Computer Aided Design), CAM (Computer Aided Manufacturing), CAGD (Computer Aided Geometric Design).

Bei den an die interpolierende Funktion φ zu stellenden Forderungen hat man zwei Eigenschaften zu unterscheiden:

a) *Interpolationseigenschaft*

$$\varphi(t_i) = f(t_i), \quad i = 1, \ldots, n$$

für gegebene Punkte $(t_i, f(t_i))$

t_i	:	Interpolationsknoten (Stützstellen)
$f(t_i)$	:	Stützwerte
$(t_i, f(t_i))$	:	Stützpunkte

Zur Erfüllung dieser Bedingung gibt es zahlreiche Möglichkeiten für die Wahl von φ. Deshalb schränkt man die Wahlmöglichkeit auf eine Funktionenklasse ϕ ein. Beispiele für in Frage kommende Funktionenklassen sind

- Polynome
- trigonometrische Funktionen
- Exponentialfunktionen
- rationale Funktionen

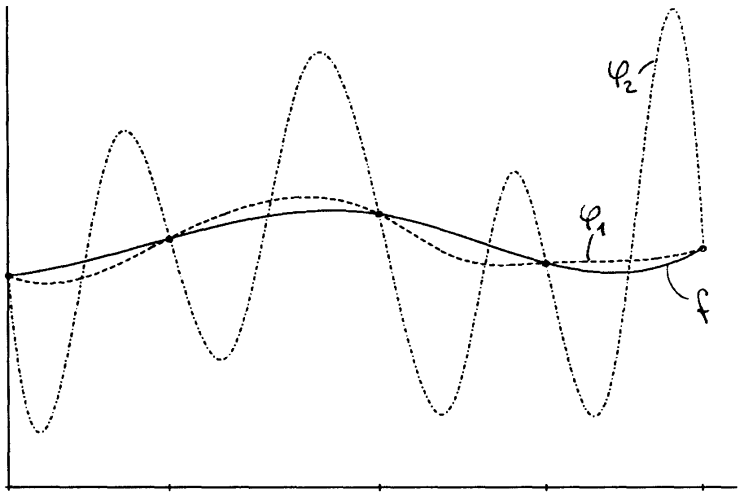


Abbildung 5.10: interpolierende Funktionen

Sehen wir uns nun graphisch die zwei verschiedenen interpolierenden Funktionen φ_1 und φ_2 von f in folgendem Bild an.

Offensichtlich genügen sowohl φ_1 als auch φ_2 der Interpolationseigenschaft. Trotzdem wird man φ_1 vorziehen. Damit wird deutlich, daß die Interpolationseigenschaft alleine nicht ausreicht, um unserer Forderung,

$$\varphi - f \quad \text{“klein”}$$

in sinnvoller Weise zu realisieren. Zusätzlich zur Interpolationseigenschaft fordert man daher noch:

b) *Approximationseigenschaft*

$$\|\varphi - f\| \quad \text{“klein”}$$

Dabei ist zum einen festzulegen, bezüglich welcher Norm der Fehler gemessen werden soll und zum anderen, was unter “klein” zu verstehen ist.

5.1 Klassische Polynom–Interpolation

Zu gegebenen Stützpunkten

$$(t_i, f_i), \quad i = 0, \ldots, n$$

bestimmt man ein Polynom $P(t)$ vom Grad $\deg P \leq n$, so daß

$$P(t_i) = f_i, \quad i = 0, \ldots, n \; .$$

Dabei setzen wir voraus, daß die Interpolationsknoten $t_0, \ldots, t_n$ paarweise verschieden sind. Sind zwei Knoten $t_i = t_j$ für $i \neq j$ gleich, so ist entweder das Problem unlösbar (falls $f_i \neq f_j$) oder wir können einen Stützpunkt weglassen (falls $f_i = f_j$). Unter dieser

Voraussetzung ist das interpolierende Polynom eindeutig bestimmt, denn falls P und Q zwei Polynome vom Grad deg P, deg $Q \leq n$ sind mit

$$P(t_i) = Q(t_i) = f_i, \quad i = 0, \ldots, n$$

so ist $P - Q$ ein Polynom maximal n-ten Grades mit den $n+1$ Nullstellen $t_0, \ldots, t_n$, also das Nullpolynom. Zur Bestimmung von P gibt es eine Reihe von Möglichkeiten.

5.1.1 Koeffizientendarstellung

Schreiben wir P in Koeffizientendarstellung

$$P(t) := a_n t^n + a_{n-1} t^{n-1} + \cdots + a_1 t + a_0$$

d.h. bezüglich der *monomialen Basis*

$$\{1, t, t^2, \ldots, t^n\}$$

des Vektorraumes

$$\mathbf{P}_n := \{P \mid P \text{ Polynom vom Grad deg } P \leq n\}$$

der Polynome maximal n-ten Grades, so lassen sich die Interpolationsbedingungen

$$P(t_i) = f_i, \quad i = 0, \ldots, n$$

als lineares Gleichungssystem

$$\underbrace{\begin{bmatrix} 1 & t_0 & t_0^2 & \ldots & t_0^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & t_n & t_n^2 & \ldots & t_n^n \end{bmatrix}}_{=:V_n} \begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f_0 \\ \vdots \\ f_n \end{bmatrix} \tag{5.1.1}$$

formulieren. Die Matrix V_n heißt *Vandermonde-Matrix*. Für die Determinante von V_n gilt

$$\det(V_n) = \prod_{i=0}^{n} \prod_{j=i+1}^{n} (t_i - t_j),$$

was in jedem Buch über lineare Algebra bewiesen wird. Daraus folgt sofort der Satz:

Satz 5.1.1 *Zu gegebenen Stützpunkten $(t_i, f_i), (i = 0, \ldots, n)$ existiert ein eindeutiges Unterpolationspolynom genau dann, wenn die Knoten $t_0, \ldots, t_n$ paarweise verschieden sind.*

Falls man $P(t)$ an vielen Zwischenstellen auswerten will, lohnt es sich, das lineare Gleichungssystem 5.1.1 zu lösen. Zur *Auswertung* eines Polynoms

$$P(t) = a_n t^n + a_{n-1} t^{n-1} + \cdots + a_1 t + a_0$$

verwendet man stets den sogenannten *Horner-Algorithmus*, der nur n Multiplikationen und n Additionen benötigt:

$$\begin{aligned} s_n &:= a_n \\ s_k &:= a_k + s_{k+1} \text{ für } k = n-1, \ldots, 0 \\ P(t) &= s_0 \end{aligned}$$

Für $n = 3$ entspricht das gerade der Ausklammerung

$$P(t) = a_0 + t(\underbrace{a_1 + z(\underbrace{a_2 + ta_3}_{s_2})}_{s_1})}_{s_0}$$

5.1.2 Die Darstellung von Lagrange

Wir können das Interpolationspolynom leichter berechnen und auswerten, wenn wir von der monomialen Basis zu anderen, dem Interpolationsproblem besser angepaßten Basen von $\mathbf{P}_n$ übergehen. Ein Beispiel dafür sind die sogenannten *Lagrange-Polynome*

$$\{L_0(t), \ldots, L_n(t)\}$$

Definition 5.1.2 *Zu gegebenen paarweise verschiedenen Stützstellen $t_0, \ldots, t_n$ ist das* i-te Lagrangepolynom $L_i(t) \in \mathbf{P}_n$ *das (eindeutig bestimmte) Interpolationspolynom mit*

$$L_i(t_j) = \delta_{ij} := \begin{cases} 0 & , \quad \textit{falls} \quad i \neq j \\ 1 & , \quad \textit{falls} \quad i = j \end{cases}$$

Man sieht sofort, daß L_i gerade das Polynom

$$L_i(t) = \frac{(t-t_0)\cdots(t-t_{i-1})(t-t_{i+1})\cdots(t-t_n)}{(t_i-t_0)\cdots(t_i-t_{i-1})(t_i-t_{i+1})\cdots(t_i-t_n)}$$

ist und daß die $n+1$ Polynome linear unabhängig sind und eine Basis von $\mathbf{P}_n$ bilden. Aus diesen Grundpolynomen $L_i(t)$ läßt sich das Interpolationspolynom für beliebige Stützwerte $f_0, \ldots, f_n$ einfach aufbauen.

$$P(t) = \sum_{i=o}^{n} f_i L_i(t)$$

Denn offensichtlich gilt

$$P(t_j) = \sum_{i=0}^{n} f_i \delta_{ij} = f_j$$

Bemerkung 5.1.3 Die obige Aussage läßt sich auch so formulieren: Die Lagrange-Polynome bilden eine Orthonormalbasis von $\mathbf{P}_n$ bezüglich des Skalarproduktes

$$< P, Q > := \sum_{i=o}^{n} P(t_i)Q(t_i)$$

für $P, Q \in \mathbf{P}_n$ und es gilt gerade

$$< P, L_i >= P(t_i)$$

und daher

$$P = \sum_{i=o}^{n} < P, L_i > L_i = \sum_{i=0}^{n} P(t_i) L_i = \sum_{i=0}^{n} f_i L_i$$

Die Lagrange-Darstellung hat im wesentlichen nur theoretischen Wert. Für die numerische Auswertung ist sie nicht geeignet.

5.1.3 Algorithmus von Aitken und Neville

Wir bezeichnen mit

$$P(t|t_{i_0}, \ldots, t_{i_k}), \quad \{i_0, \ldots, i_k\} \subset \{0, \ldots, n\}$$

($i_0, \ldots, i_k$ paarweise verschieden) das Interpolationspolynom zu den Stützpunkten

$$(t_{i_0}, f_{i_0}), \ldots, (t_{i_k}, f_{i_k}),$$

d.h. $P(t|t_{i_0}, \ldots, t_{i_k})$ ist das Polynom vom Grad deg $\leq k$ mit

$$P(t_\ell|t_{i_0}, \ldots, t_{i_k}) = f_\ell \text{ für } \ell = i_0, \ldots, i_k \, .$$

Diese Polynome lassen sich rekursiv berechnen. Betrachtet man die zwei "Nachbar"-Polynome (vom Grad $\leq k-1$)

$$P(t|t_{i_0}, t_{i_1}, \ldots, t_{i_{k-1}}) \text{ und } P(t|t_{i_1}, \ldots, t_{i_{k-1}}, t_{i_k})$$

so gilt:

Satz 5.1.4 *(AITKEN) Es gilt die Rekursionsformel*

$$P(t|t_{i_0}, \ldots, t_{i_k}) = \frac{(t_{i_0} - t)P(t|t_{i_1}, \ldots, t_{i_k}) + (t - t_{i_k})P(t|t_{i_0}, \ldots, t_{i_{k-1}})}{t_{i_o} - t_{i_k}} \tag{5.1.2}$$

Beweis: Sei $\varphi(t)$ definiert als der Ausdruck auf der rechten Seite von 5.1.2. φ ist vom Grad deg $\varphi \leq k$ und für die gemeinsamen Knoten der Nachbarpolynome

$$t_\ell, \quad \ell = i_1, \ldots, i_{k-1}$$

gilt

$$\varphi(t_\ell) = \frac{(t_{i_0} - t_\ell)f_\ell + (t_\ell - t_{i_k})f_\ell}{t_{i_0} - t_{i_k}} = f_\ell$$

Ebenso leicht folgt

$$\varphi(t_{i_0}) = f_{i_0} \text{ und } \varphi(t_{i_k}) = f_{i_k}$$

Also ist φ das Interpolationspolynom zu den Punkten

$$(t_{i_0}, f_{i_0}), \ldots, (t_{i_k}, f_{i_k}) \, .$$

□

Da die Interpolationspolynome für nur einen Stützpunkt nichts weiter als die Konstanten

$$P(t,t_i) = f_i, \quad i = 0,\ldots,n$$

sind, läßt sich $P = P(t|t_0,t_1,\ldots,t_n)$ rekursiv nach dem *Schema von Neville* berechnen:

$$\begin{array}{lccccccc}
f_0 = P(t|t_0) & & & & & & & \\
& \searrow & & & & & & \\
f_1 = P(t|t_1) & \rightarrow & P(t|t_0,t_1) & & & & & \\
\vdots & & \vdots & & & & & \\
\vdots & & \vdots & & & & & \\
f_{n-1} = P(t|t_{n-1}) & \rightarrow & \cdots & \rightarrow & P(t|t_0,\ldots,t_{n-1}) & & & \\
& \searrow & \cdots & & & \searrow & & \\
f_n = P(t|t_n) & \rightarrow & \cdots & \rightarrow & P(t|t_1,\ldots,t_n) & \rightarrow & P(t|t_0,\ldots,t_n)
\end{array}$$

Vereinfacht man die Indizierung durch

$$P_{ik}(t) := P(t \mid t_{i-k},\ldots,t_i) \text{ für } i \geq k$$

so berechnet man

$$P_{nn}(t) = P(t|t_0,\ldots,t_n)$$

für ein gegebenes festes t und Stützpunkte (t_i, f_i) $(i = 0,\ldots,n)$ gemäß dem Schema

$$\begin{array}{lccccccc}
P_{00} & & & & & & & \\
& \searrow & & & & & & \\
P_{10} & \rightarrow & P_{11} & & & & & \\
& \searrow & & \searrow & & & & \\
P_{20} & \rightarrow & P_{21} & \rightarrow & P_{22} & & & \\
\vdots & & & & \vdots & & & \\
\vdots & & & & \vdots & & & \\
P_{n-1,0} & \rightarrow & \cdots & \rightarrow & P_{n-1,n-1} & & & \\
& \searrow & & & & \searrow & & \\
P_{n0} & \rightarrow & \cdots & \rightarrow & P_{n,n-1} & \rightarrow & P_{nn}
\end{array} \tag{5.1.3}$$

nach der Vorschrift

$$\begin{array}{ll}
\text{a)} & P_{i0} = f_i \\
\text{b)} & P_{ik} = P_{i,k-1} + \dfrac{t - t_i}{t_i - t_{i-k}}(P_{i,k-1} - P_{i-1,k-1})
\end{array} \tag{5.1.4}$$

Dies läßt sich mit einem einzigen n-Feld realisieren.

Beispiel 5.1.5 *Berechnung von* $\sin 62^\circ$ *aus den Stützpunkten*

$$(50^\circ, \sin\ 50^\circ), (55^\circ, \sin\ 55^\circ), \ldots, (70^\circ, \sin\ 70^\circ)$$

mit dem Aitken-Neville-Algorithmus

t_i	$\sin\ t_i$				
50°	0.7660444				
55°	0.8191520	0.8935026			
60°	0.8660254	0.8847748	0.8830292		
65°	0.9063078	0.8821384	0.8829293	0.8829493	
70°	0.9396926	0.8861384	0.8829661	0.8829465	0.8829476

An diesem Beispiel erkennt man, daß sich an den bei der rekursiven Berechnung auftretenden Zahlen ablesen läßt (hier durch Abzählen der gültigen Ziffern) bis zu welcher Ordnung, d.h. mit wievielen Stützpunkten interpoliert werden muß, um eine bestimmte Genauigkeit zu erreichen. Diesen Vorteil werden wir bei der automatischen Ordnungssteuerung bei Extrapolationsmethoden wie z.B. der Romberg-Quadratur (Kapitel 6.3) schätzen lernen.

Der Aufwand beträgt dabei

$$\sim \frac{1}{2}n^2 \text{ Division und } \sim \frac{1}{2}n^2 \text{ Multiplikationen}$$

Der Algorithmus wird hauptsächlich im Spezialfall $t = 0$ angewandt, wodurch sich (.) zu

$$\begin{aligned} P_{ik} &= P_{i,k-1} - \frac{t_i}{t_i - t_{i-k}}(P_{i,k-1} - P_{i-1,k-1}) \\ &= P_{i,k-1} + \frac{(P_{i,k-1} - P_{i-1,k-1})}{t_{i-k}/t_i - 1} \end{aligned} \tag{5.1.5}$$

vereinfacht und der Quotient für verschiedene Knoten t_i nur einmal berechnet werden muß.

5.1.4 Newton'sche dividierte Differenzen

Soll das Interpolationspolynom an vielen Stellen ausgewertet werden, so ist das folgende auf Newton zurückgehende Verfahren günstiger. Dabei verwenden wir als Basis von $\mathbf{P}_n$ die Polynome

$$w_k(t) := \prod_{i=0}^{k-1}(t - t_i) = \begin{cases} 1 & , \text{ falls } k = 0 \\ (t - t_0)\cdots(t - t_{k-1}) & , \text{ falls } k = 1, \ldots, n \end{cases}$$

Die Koeffizienten $\alpha_0, \ldots, \alpha_n$ eines Polynoms $P \in \mathbf{P}_n$,

$$P(t) = \sum_{k=0}^{n} \alpha_k w_k(t) = \alpha_0 + \alpha_1(t - t_0) + \cdots + \alpha_n(t - t_0)\cdots(t - t_{n-1}),$$

können wir sukzessive aus den Funktionswerten $P(t_i)$ an den Stützstellen $t_0, \ldots, t_n$ ermitteln.

$$\begin{aligned} P(t_0) &= \alpha_0 \\ P(t_1) &= \alpha_0 + \alpha_1(t - t_0) \\ &\vdots \\ P(t_n) &= \alpha_0 + \alpha_1(t_n - t_0) + \cdots + \alpha_n(t_n - t_0)\cdots(t_n - t_{n-1}) \end{aligned}$$

Ist $P(t) = P(t|t_0, \ldots, t_n)$ das Interpolationspolynom zu den Stützpunkten (t_i, f_i), so ist das Interpolationspolynom zu den ersten $k+1$ Stützpunkten $(k = 0, \ldots, n)$ gerade

$$\begin{aligned} P(t|t_0, \ldots, t_k) &= \sum_{i=0}^{k} \alpha_i w_i(t) \\ &= \alpha_0 + \alpha_1(t - t_0) + \cdots + \alpha_k(t - t_0)\cdots(t - t_{k-1}) \end{aligned}$$

und der Koeffizient α_k des Interpolationspolynoms $P(t|t_0, \ldots, t_n)$ bezüglich der Basis $\{w_i(t)\}$ ist der Koeffizient von t^k des Polynoms $P(t|t_0, \ldots, t_k)$. Zusammen mit dem Satz von Aitken erhält man anstelle von (.) eine rekursive Methode zur Bestimmung der Koeffizienten $\alpha_0, \ldots, \alpha_n$, die wir in folgendem Satz formulieren.

Satz 5.1.6 *Definieren wir rekursiv für $i = 0, \ldots, n$ und $k = 1, \ldots, n - i$*

a) $f[t_i] := f_i$

b) $f[t_i, \ldots, t_{i+k}] := \dfrac{f[t_{i+1}, \ldots, t_{i+k}] - f[t_i, \ldots, t_{i+k-1}]}{t_{i+k} - t_i}$

c) $\alpha_i := f[t_0, \ldots, t_i]$

so ist das Polynom

$$P(t) := \sum_{i=0}^{n} \alpha_i w_i(t) \in \mathbf{P}_n$$

das Interpolationspolynom

$$P(t) = P(t|t_0, \ldots, t_n)$$

zu den Stützpunkten $(t_0, f_0), \ldots, (t_n, f_n)$. □

Die Quotienten in b) heißen *dividierte Differenzen* und geben dem Verfahren seinen Namen. Sie lassen sich wie beim Verfahren von Neville in einem Schema

$$\begin{array}{lcccccc} f_0 = f[t_0] & & & & & & \\ & \searrow & & & & & \\ f_1 = f[t_1] & \rightarrow & f[t_0, t_1] & & & & \\ \vdots & & \vdots & & & & \\ \vdots & & \vdots & & & & \\ f_{n-1} = f[t_{n-1}] & \rightarrow & \cdots & \rightarrow & f[t_0, \ldots, t_{n-1}] & & \\ & \searrow & \cdots & & & \searrow & \\ f_n = f[t_n] & \rightarrow & \cdots & \rightarrow & f[t_1, \ldots, t_n] & \rightarrow & f[t_0, \ldots, t_n] \end{array}$$

anordnen, wobei sich die Koeffizienten α_i aus der obersten Schrägzeile ablesen lassen. Hat man die Koeffizienten α_i bestimmt, so läßt sich das Interpolationspolynom leicht mit Hilfe einer Art Horner-Schema an einer Stelle t auswerten

$$\begin{aligned} s_n &:= \alpha_n \\ s_k &:= a_k + s_{k+1}(t - t_k) \text{ für } k = n-1, \ldots, 0 \\ P(t) &= s_0 \end{aligned}$$

Beispiel 5.1.7 *Wir berechnen das Interpolationspolynom zu den Werten*

t_i	0	1	2	3
f_i	1	2	0	1

mit Hilfe der Newton'schen dividierten Differenzen.

$$\begin{array}{llll} f[t_0] = 1 & & & \\ f[t_1] = 2 & f[t_0, t_1] = 1 & & \\ f[t_2] = 0 & f[t_1, t_2] = -2 & f[t_0, t_1, t_2] = -3/2 & \\ f[t_3] = 1 & f[t_2, t_3] = 1 & f[t_1, t_2, t_3] = 3/2 & f[t_0, t_1, t_2, t_3] = 1 \end{array}$$

also

$$(\alpha_0, \alpha_1, \alpha_2, \alpha_3) = (1, 1, -3/2, 1)$$

Das Interpolationspolynom ist daher

$$\begin{aligned} P(t) &= 1 + 1(t-0) + (-3/2)(t-0)(t-1) + 1(t-0)(t-1)(t-2) \\ &= t^3 - 4.5\, t^2 + 4.5\, t + 1 \end{aligned}$$

□

5.1.5 Approximationsfehler

Nachdem wir nun das Interpolationspolynom $P(t) = P(t|t_0, \ldots, t_n)$ zu gegebenen Stützpunkten $(t_0, f_0), \ldots, (t_n, f_n)$ gut berechnen können, stellt sich die Frage, inwieweit wir damit auch unsere zweite Forderung, die Approximationseigenschaft,

$$\|f - P\| \text{ "klein"}$$

für eine gegebene Funktion f erfüllt haben. Dazu leiten wir in dem folgenden Satz einen Ausdruck für den *Approximationsfehler*

$$f(\bar{t}) - P(\bar{t})$$

an einer festen Stelle $\bar{t}$ ab.

Satz 5.1.8 *Sei $[a,b]$ ein Intervall, das die Stützstellen $t_0,\ldots,t_n$ und ein festes $\bar{t}$ enthält. Sei weiter $f \in C^{n+1}[a,b]$ und $P_n(t) \in \mathbf{P}_n$ das Interpolationspolynom vom Grad* deg $P_n \leq n$ *über den Stützpunkten $(t_0,f(t_0)),\ldots,(t_n,f(t_n))$. Dann gibt es ein $\tau \in (a,b)$, abhängig von $\bar{t}$, so daß*

$$f(\bar{t}) - P_n(\bar{t}) = \frac{f^{(n+1)}(\tau)}{(n+1)!}(\bar{t}-t_0)\cdots(\bar{t}-t_n)$$

Beweis: Falls $\bar{t} = t_i$ für ein $i \in \{0,\ldots,n\}$, ist die Behauptung klar, so daß wir davon ausgehen können, daß $\bar{t} \neq t_i$ für $i = 0,\ldots,n$. Dann ist

$$w(\bar{t}) := (\bar{t}-t_0)\cdots(\bar{t}-t_n) \neq 0$$

Für die Funktion

$$F(t) := f(t) - P_n(t) - Kw(t)$$

gilt für jedes beliebige $K \in \mathbf{R}$, daß $F \in C^{n+1}[a,b]$ und

$$F(t_i) = 0 \text{ für } i = 0,\ldots,n.$$

Setzen wir speziell

$$K := \frac{f(\bar{t}) - P_n(\bar{t})}{w(\bar{t})}$$

so folgt $F(\bar{t}) = 0$, d.h. F hat $n+2$ Nullstellen in $[a,b]$. Nach dem Satz von Rolle hat daher $F^{(n+1)}$ mindestens eine Nullstelle τ in (a, b), also

$$\begin{aligned} 0 &= F^{(n+1)}(\tau) \\ &= f^{(n+1)}(\tau) - kw^{(n+1)}(\tau) \\ &= f^{(n+1)}(\tau) - \frac{f(\bar{t}) - P_n(\bar{t})}{w(\bar{t})}(n+1)! \end{aligned}$$

und daher wie verlangt

$$f(\bar{t}) - P_n(\bar{t}) = \frac{f^{(n+1)}(\tau)}{(n+1)!}w(\bar{t}) \ .$$

□

Offenbar hängt der Approximationsfehler stark von der Wahl der Stützstellen $t_0,\ldots,t_n$ ab. Betrachten wir die Klasse der Funktionen mit durch eine Konstante $M \geq 0$ beschränkter $(n+1)$-ter Ableitung

$$\mathcal{F} := \{f \in C^{n+1}[a,b] \mid \ |f^{(n+1)}(\tau)| \leq M, \ \ \tau \in [a,b]\}$$

so können wir zur Minimierung des Approximationsfehlers diejenigen Stützknoten $t_0,\ldots,t_n$ bestimmen, für die

$$\max_{t\in[a,b]} |w(t)| \ = \ min.$$

Dies führt zu den sogenannten *Tschebyscheff-Knoten*, die für das Intervall $[a,b] = [-1,1]$ gegeben sind durch

$$t_i := \cos(\pi \cdot \frac{i+1}{n+2}) \text{ für } i = 0,\ldots,n$$

In der Praxis sind die Stützstellen jedoch meistens vorgegeben, so daß diese Überlegungen wenig weiterhelfen.

Kommen wir zurück zu der Frage, ob die Polynominterpolation die Approximationseigenschaft erfüllt. Für stetige Funktionen $f \in C[a,b]$ und die Supremumsnorm $\|f\| = \sup_{t\in[a,b]} f(t)$ lautet die Antwort im allgemeinen nein. Genauer existiert nach FABER für jede Folge $\{T_k\}$ Stützstellensätzen $T_k = \{t_{k,0},\ldots,t_{k,n_k}\} \subset [a,b]$ eine stetige Funktion $f \in C[a,b]$, so daß die Folge $\{P_k\}$ der zu den T_k gehörenden Interpolationspolynome nicht gleichmäßig gegen f konvergiert.

Bemerkung: Liegt $\bar{t}$ außerhalb des kleinsten Intervalls, das alle Knoten t_i enthält, so sprechen wir auch von *Extrapolation.* Bei der *Polynomextrapolation* wertet man einfach das Interpolationspolynom $P_n(t)$ an der Stelle $\bar{t}$ aus.

5.1.6 Kondition

Eingabedaten bei der Interpolation sind die Knoten $\{t_i\}$ und die Stützwerte $\{f_i\}$. Betrachten wir die Auswirkung von Störungen der Stützwerte $f_0,\ldots,f_n$ auf den Wert $P(t)$ des Interpolationspolynoms an einer festen Stelle t, d.h. in der Notation des zweiten Kapitels

Eingabedaten	:	$f_0,\ldots,f_n$
Resultat	:	$P(t)$ für ein festes t
Störung der Eingabedaten	:	$f_i \to f_i + \delta f_i$
Störung des Resultats	:	$P \to p + \delta P$

so erhalten wir mit den Lagrange-Polynomen $L_0(t),\ldots,L_n(t)$ die Formel

$$\delta P(t) = \sum_{i=0}^{n} \delta f_i L_i(t) \; .$$

Die Beträge $|L_i(t)|$ der Lagrange-Polynome lassen sich daher als *absolute Konditionszahlen* bzgl. Störungen der f_i interpretieren.

Beispiel 5.1.9 *Polynominterpolation mit den äquidistanten Stützstellen $t_i = i$ und Stützwerten $f_i = \delta_{i,m}$ für $i = 0,\ldots,2m =: n$, d. h. $P(t) = L_m(t)$.*

m	$L_m(0.5)$
0	1
5	-5
10	-10^3
20	-3.10^8
50	-6.10^{25}

Zusammenfassung:

Die klassische Polynominterpolation hoher Ordnung ist bei äquidistanten Knoten schlecht konditioniert, bei Tschebyscheff-Knoten jedoch gut konditioniert.

5.1.7 Hermite-Interpolation

Bisher haben wir an den Knoten $t_0, \ldots, t_n$ die Funktionswerte $f_0, \ldots, f_n$ vorgegeben und das dadurch eindeutig bestimmte Interpolationspolynom vom Grad n berechnet. Genauso ist es möglich, zusätzlich die Werte der Ableitungen an den Knoten t_i bis zu einem Grad s_i festzulegen und damit ein Polynom entsprechend höheren Grades zu charakterisieren.

Satz 5.1.10 *Gegeben seien $n+1$ paarweise verschiedene Stützstellen $t_0, \ldots, t_n$ und Daten $f_i^{(j)}$ für $i = 0, \ldots, n$ und $j = 0, \ldots, s_i$. Dann existiert genau ein Polynom*

$$P \in \mathbf{P}_m \quad \textit{mit} \quad m := n + \sum_{i=0}^{n} s_i$$

derart, daß

$$P^{(j)}(t_i) = f_i^{(j)} \quad \textit{für } i = 0, \ldots, n \textit{ und } j = 0, \ldots, s_i$$

Definition 5.1.11 *Das eindeutig bestimmte Polynom $P \in \mathbf{P}_m$ heißt* Hermite-Polynom *zu den Werten $\{t_i, f_i^{(j)}\}$.*

Beweis: Wir zeigen zunächst die Eindeutigkeit (vgl. Anfang von Kapitel 5.1). Sind $P, Q \in \mathbf{P}_m$ zwei Polynome mit

$$P^{(j)}(t_i) = Q^{(j)}(t_i) = f_i^{(j)} \text{ für } i = 0, \ldots, n \text{ und } j = 0, \ldots, s_i$$

so ist $P - Q \in \mathbf{P}_m$ ein Polynom mit

$$(P-Q)^{(j)}(t_i) = 0 \text{ für } i = 0, \ldots, n \text{ und } j = 0, \ldots, s_i$$

und daher $P - Q = 0$. Das Hermite-Polynom ist also eindeutig bestimmt. Daraus folgt auch die Existenz (ausnahmsweise einmal nicht konstruktiv), denn die Abbildung

$$\mathbf{P}_m \longrightarrow \mathbf{R}^{m+1}, \quad P \longmapsto (P^{(j)}(t_i))_{\substack{i=0,\ldots,n \\ j=0,\ldots,s_i}}$$

ist linear und injektiv, also auch surjektiv. □

Bei der *kubischen Hermite-Interpolation* gibt man an zwei Knoten t_0, t_1 jeweils Funktionswert und Ableitung vor und berechnet das dadurch eindeutig bestimmte Polynom $P \in \mathbf{P}_3$. Definiert man die Hermite-Polynome $H_i^3(t) \in \mathbf{P}_3$, $i = 0, \ldots, 3$, durch

$$\begin{aligned}
&H_0^3(t_0) = 1, \quad \frac{d}{dt}H_0^3(t_0) = 0, \quad H_0^3(t_1) = 0, \quad \frac{d}{dt}H_0^3(t_1) = 0,\\
&H_1^3(t_0) = 0, \quad \frac{d}{dt}H_1^3(t_0) = 1, \quad H_1^3(t_1) = 0, \quad \frac{d}{dt}H_1^3(t_1) = 0,\\
&H_2^3(t_0) = 0, \quad \frac{d}{dt}H_2^3(t_0) = 0, \quad H_2^3(t_1) = 1, \quad \frac{d}{dt}H_2^3(t_1) = 0,\\
&H_3^3(t_0) = 0, \quad \frac{d}{dt}H_3^3(t_0) = 0, \quad H_3^3(t_1) = 0, \quad \frac{d}{dt}H_3^3(t_1) = 1,
\end{aligned}$$

so bilden die Polynome

$$\{H_0^3(t), H_1^3(t), H_2^3(t), H_3^3(t)\}$$

eine Basis von $\mathbf{P}_3$, die sogenannte *kubische Hermite-Basis* bzgl. der Knoten t_0, t_1. Das Hermite-Polynom zu den Werten $\{f_0, f_0', f_1, f_1'\}$ läßt sich damit formal einfach angeben durch

$$P(t) = f_0 H_0^3(t) + f_0' H_1^3(t) + f_1 H_2^3(t) + f_1' H_3^3(t).$$

Wir kommen auf die Polynome $H_i^3(t)$ im Kapitel 5.3 zurück.

5.2 Kubische Spline-Interpolation

Wie wir gesehen haben, ist die klassische Polynominterpolation ungeeignet, das Approximationsproblem bei einer großen Anzahl von äquidistanten Knoten zu lösen. Polynome hohen Grades neigen zu starken Oszillationen, wie die Skizzen der Lagrange-Polynome verdeutlichen. Sie verderben so evtl. nicht nur die Kondition (kleine Änderungen der Stützwerte f_i bewirken große Änderungen des Interpolationspolynoms $P(t)$ an Zwischenstellen $t \neq i_i$), sondern führen auch zu starken Schwingungen der interpolierenden Kurve zwischen den Stützstellen. Wie man sich vorstellen kann, sind derartige Schwingungen höchst unerwünscht. Man denke nur an die dadurch verursachten Vibrationen einer Flugzeugtragfläche, die nach einer solchen Interpolationskurve geformt wurde. Verlangen wir, daß eine interpolierende Kurve "möglichst glatt" durch gegebene Stützpunkte $(t_0, f_0, \ldots, (t_n, f_n)$ verläuft, so liegt es näher, lokal Polynome *niedrigen Grades* zu verwenden und diese an den Stützpunkten miteinander zu verheften. Dies führt zu den sogenannten *Spline-Funktionen.*

Definition 5.2.1 *Ein* Spline vom Grad k *bzgl. der Knoten* $t_0 < \cdots < t_n \in \mathbf{R}$ *ist eine Funktion* $s \in C^{k-1}[t_0, t_n]$, *die auf jedem Intervall* $[t_i, t_{i+1}]$, $i = 0, \ldots, n-1$, *mit einem Polynom* $s_i \in \mathbf{P}_k$ *vom Grad* k *übereinstimmt.*

5.2.1 Berechnung kubischer Splines

Wir werden uns im folgenden ausschließlich mit Splines vom Grad 3, den sogenannten *kubischen Splines* beschäftigen. Diese eignen sich bestens zur graphischen Darstellung von Kurven, da das Auge Unstetigkeiten in der Krümmung (also der zweiten Ableitung) noch erkennen kann. Damit werden die C^2-stetigen kubischen Splines als "glatt" wahrgenommen.

Zur Darstellung eines kubischen Splines $s(t)$ schreiben wir die Polynome $s_i \in \mathbf{P}_3$ als

$$s_i(t) = a_i + b_i(t - t_i) + \frac{c_i}{2}(t - t_i)^2 + \frac{d_i}{6}(t - t_i)^3 \qquad (5.2.1)$$

d.h. bezüglich der Basispolynome $1, t - t_i, (t - t_i)^2$ und $(t - t_i)^3$. Der Spline s ist daher charakterisiert durch die $4n$ Koeffizienten

$$(a_i, b_i, c_i, d_i) \text{ für } i = 0, \ldots, n$$

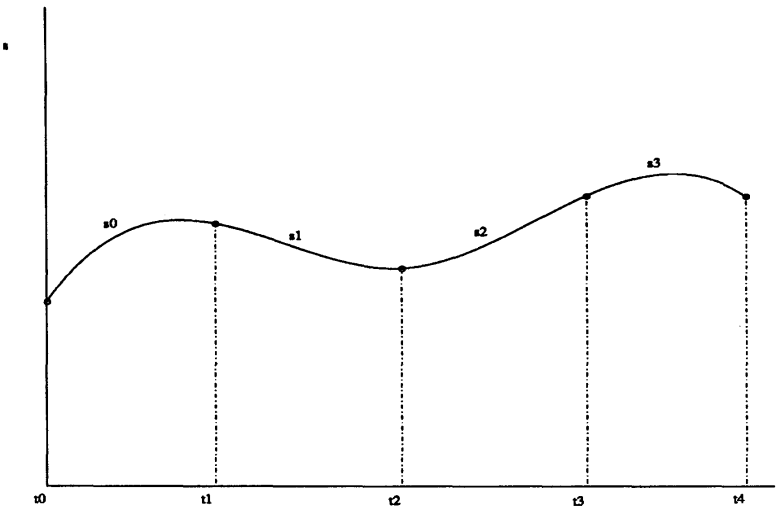


Abbildung 5.11: kubischer Spline mit 5 Stützpunkten

und es gilt

$$\begin{aligned} \text{a)} \quad s_i'(t) &= b_i + c_i(t-t_i) + \tfrac{d_i}{2}(t-t_i)^2 \\ \text{b)} \quad s_i''(t) &= c_i + d_i(t-t_i) \end{aligned}$$

Bezeichnen wir die Längen der Teilintervalle mit

$$h_i := t_{i+1} - t_i \quad \text{für} \quad i = 0, \ldots, n-1,$$

so folgt insbesondere für die Funktionswerte und Ableitungen an den Knoten t_i, daß

$$\begin{aligned} s_i(t_i) = a_i \quad &, \quad s_i(t_{i+1}) = a_i + b_i h_i + \frac{c_i}{2}h_i^2 + \frac{d_i}{6}h_i^2 \\ s_i'(t_i) = b_i \quad &, \quad s_i'(t_{i+1}) = b_i + c_i h_i + \frac{d_i}{2}h_i^2 \\ s_i''(t_i) = c_i \quad &, \quad s_i''(t_{i+1}) = c_i + d_i h_i \end{aligned} \tag{5.2.2}$$

Damit s C^2-stetig ist, müssen Funktionswerte und erste und zweite Ableitungen zweier aufeinanderfolgender Polynome s_i, s_{i+1} an der Nahtstelle t_{i+1} übereinstimmen, d.h. es müssen die sogenannten *Verheftungsbedingungen*

$$\begin{aligned} \text{a)} \quad s_i(t_{i+1}) &= s_{i+1}(t_{i+1}) \\ \text{b)} \quad s_i'(t_{i+1}) &= s_{i+1}'(t_{i+1}) \quad \text{für} \quad i = 0, \ldots, n-2 \\ \text{c)} \quad s_i''(t_{i+1}) &= s_{i+1}''(t_{i+1}) \end{aligned} \tag{5.2.3}$$

erfüllt sein. Setzt man die Gleichungen 5.2.2 ein, so folgt:

Lemma 5.2.2 *Die Verheftungsbedingungen 5.2.3 sind äquivalent zu*

(i) $\quad d_i = \Delta c_i = \dfrac{c_{i+1} - c_i}{h_i}$ *für* $i = 0, \ldots, n-2$

(ii) $\quad b_i = \Delta a_i - \frac{h_i}{6}(c_{i+1} + 2c_i)$ *für* $i = 0, \ldots, n-2$

(iii) $\quad b_{n-1} = \Delta a_{n-2} + \frac{h_{n-2}}{6}(2c_{n-1} + c_{n-2})$

(iv) $\quad \frac{h_{i-1}}{h_i + h_{i-1}} c_{i-1} + 2c_i + \frac{h_i}{h_i + h_{i-1}} c_{i+1} = \frac{6(\Delta a_i - \Delta a_{i-1})}{h_i + h_{i-1}}$ *für* $i = 1, \ldots, n-2,$

wobei Δ *der "Differenzenquotientenoperator"*

$$\Delta x_i := \frac{x_{i+1} - x_i}{t_{i+1} - t_i} = \frac{x_{i+1} - x_i}{h_i}$$

Beweis: Einfaches Einsetzen von 5.2.2 in 5.2.3 ergibt

$$\begin{aligned}
\text{a)} \quad &\Longleftrightarrow \quad a_{i+1} = a_i + b_i h_i + \frac{c_i}{2} h_i^2 + \frac{d_i}{6} h_i^3 \\
&\overset{(i)}{\Longrightarrow} \quad \Delta a_i = b_i + \frac{c_i}{2} h_i + \frac{c_{i+1} - c_i}{6} h_i^2 \Longleftrightarrow (ii) \\
\text{b)} \quad &\Longleftrightarrow \quad b_i = b_{i-1} + c_{i-1} h_{i-1} + \frac{d_{i-1}}{2} h_{i-1}^2 \\
&\overset{(ii)}{\Longrightarrow} \quad b_i = \Delta a_{i-1} - \frac{h_{i-1}}{6}(c_i + 2c_{i-1}) + c_{i-1} h_{i-1} + \frac{c_i - c_{i-1}}{2} h_{i-1} \\
&\Longleftrightarrow \quad b_i = \Delta a_{i-1} + \frac{h_{i-1}}{6}(2c_i + c_{i-1}) \\
\text{c)} \quad &\Longleftrightarrow \quad c_{i+1} = c_i + d_i h_i \\
&\Longleftrightarrow \quad (i)
\end{aligned}$$

Für $i = n-1$ ist dies gerade (iii) und Einsetzen von (ii) auf der linken Seite liefert (iv). □

Wollen wir den *interpolierenden Spline* zu den Stützpunkten $(t_0, f_0), \ldots, (t_n, f_n)$ bestimmen, so kommen $n+1$ *Interpolationsbedingungen* hinzu:

$$s(t_i) = f_i \quad \text{für} \quad i = 0, \ldots, n\,.$$

Zusammen mit den Verheftungsbedingungen haben wir damit $4n-2$ Gleichungen, um die $4n$ unbekannten Koeffizienten $\{a_i, b_i, c_i, d_i\}$ zu bestimmen. Die zwei noch freien Wünsche werden durch die *Randbedingungen* festgelegt. Dabei unterscheidet man drei Typen, die in Kapitel 5.2.2 begründet werden.

Typ I	:	$s''(t_0) = s''(t_n) = 0$ (*natürliche Randbedingung*)
Typ II	:	$s'(t_0) = f_0'$, $s'(t_n) = f_n'$ zusätzlich vorgegeben (*eingespannter Rand*)
Typ III	:	$s'(t_0) = s'(t_n)$, $s''(t_0) = s''(t_n)$ (*periodische Randbedingung*)

Dabei ist der dritte Typ nur sinnvoll, falls die zu approximierende Funktion periodisch ist, d.h. $f(t) = f(t + k(t_n - t_0))$ für $k \in \mathbf{Z}$. Wir führen die Rechnungen hier nur für die ersten beiden Typen aus.

Satz 5.2.3 *1) Für die Typen I, II und III (nur falls $f_0 = f_n$) existiert genau ein interpolierender kubischer Spline, der den entsprechenden Randbedingungen genügt.*

2) Für Typ I lassen sich die Koeffizienten durch

$$\begin{aligned} &(i) && a_i := f_i \ \text{für}\ i = 0,\dots,n-1 \\ &(ii) && d_i := \Delta c_i \ \text{für}\ i = 0,\dots,n-1 \\ &(iii) && b_i := \Delta f_i - \frac{h_i}{6}(c_{i+1} + 2c_i) \ \text{für}\ i = 0,\dots,n-1 \end{aligned} \tag{5.2.4}$$

berechnen, wobei $c_n := c_0 := 0$ und $c_1,\dots,c_{n-1}$ die Lösungen des linearen Gleichungssystems

$$\underbrace{\begin{bmatrix} 2 & \lambda_1 & & & \\ \mu_2 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \lambda_{n-2} \\ & & & \mu_{n-1} & 2 \end{bmatrix}}_{=:\, M_I \in Mat_{n-1}(\mathbf{R})} \underbrace{\begin{pmatrix} c_1 \\ \vdots \\ \vdots \\ \vdots \\ c_{n-1} \end{pmatrix}}_{=:\, c \in \mathbf{R}^{n-1}} = \underbrace{\begin{pmatrix} \delta_1 \\ \vdots \\ \vdots \\ \vdots \\ \delta_{n-1} \end{pmatrix}}_{=:\, \delta \in \mathbf{R}^{n-1}} \tag{5.2.5}$$

mit

$$\mu_i := \frac{h_{i-1}}{h_i + h_{i-1}}, \quad \lambda_i := \frac{h_i}{h_i + h_{i-1}}, \quad \delta_i := \frac{6(\Delta f_i - \Delta f_{i-1})}{h_i + h_{i-1}} \tag{5.2.6}$$

sind.

3) Für Typ II berechnen sich die Koeffizienten mit den Gleichungen 5.2.4 wie beim Typ I, wobei $c_0,\dots,c_n$ die Lösungen des Gleichungssystems

$$\begin{bmatrix} 2 & \lambda_0 & & & \\ \mu_1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \lambda_{n-1} \\ & & & \mu_n & 2 \end{bmatrix} \begin{pmatrix} c_0 \\ \vdots \\ \vdots \\ \vdots \\ c_n \end{pmatrix} \begin{pmatrix} \delta_0 \\ \vdots \\ \vdots \\ \vdots \\ \delta_n \end{pmatrix}$$

und zusätzlich zu 5.2.6

$$\begin{aligned} \lambda_0 &:= 1, \ \mu_n := 1, \\ \delta_0 &:= \frac{6(\Delta f_0 - f_0')}{h_0}, \quad \delta_n := \frac{6(f_n' - \Delta f_{n-1})}{h_{n-1}} \end{aligned}$$

gesetzt wird.

Beweis: Die Gleichungen ?? folgen sofort aus den Interpolationsbedingungen

$$s(t_i) = s_i(t_i) = a_i \ \text{für}\ i = 0,\dots,n-1$$

und Lemma 5.2.2 (i) bis (iii). Das Gleichungssystem 5.2.5 entspricht Lemma 5.2.2 (iv). Wir müssen daher nur noch die Randbedingungen einarbeiten und beschränken uns dabei auf die ersten beiden Typen. Mit $c_n := s'(t_n)$, $b_n := s'(t_n)$ folgt dann:

Typ I: $c_0 = s_0''(t_0) = s''(t_0) = 0$ und $c_n = s''(t_n) = 0$

Typ II: Es sind $b_0 = f_0'$ und $b_n = f_n'$ vorgegeben und wie in Lemma 5.2.2 gilt

$$f_0' = \Delta f_0 - \frac{h_0}{6}(c_1 + 2c_0) \iff 2c_0 + \underbrace{c_1}_{\Longrightarrow \lambda_0 := 1} = \frac{6(\Delta f_0 + f_0')}{h_0} =: \delta_0$$

sowie

$$f_n' = \Delta f_{n-1} + \frac{h_{n-1}}{6}(2c_n + c_{n-1}) \iff \underbrace{c_{n-1}}_{\Longrightarrow \mu_n := 1} + 2c_n = \frac{6(f_n' - \Delta f_{n-1})}{h_{n-1}} =: \delta_n$$

Für die eindeutige Lösbarkeit bleibt nur noch zu zeigen, daß die Matrizen M_I und M_{II} nicht singulär sind. Da

$$|\lambda_i\| + \|\mu_i| = 1 < 2$$

folgt, daß der Kern von M_I und M_{II} nulldimensional ist. □

Bemerkung 5.2.4 Die Matrizen M_I und M_{II} sind sogar *diagonaldominante Triagonalmatrizen*, d.h. sie besitzen nur von Null verschiedene Komponenten auf der Haupt- und den beiden Nebendiagonalen und erfüllen das *starke Spaltensummenkriterium*

$$\sum_{\substack{i=1 \\ i \neq k}}^{n} |a_{ik}| < |a_{kk}| \quad \text{für} \quad k = 1, \ldots, n$$

(für eine Matrix $A \in Mat_n(\mathbf{R})$). Derartige Matrizen besitzen ähnlich wie die symmetrisch positiv definiten sehr günstige numerische Eigenschaften. So ist die Gauß-Elimination ohne Pivotsuche numerisch stabil, was die Bandstruktur der Matrizen zu erhalten erlaubt.

Bemerkung 5.2.5 Für äquidistante Stützstellen

$$t_i := t_0 + ih, \quad h := \frac{t_n - t_0}{n}$$

vereinfachen sich die Matrizen zu

$$M_I = \begin{bmatrix} 2 & \frac{1}{2} & & & \\ \frac{1}{2} & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \frac{1}{2} \\ & & & \frac{1}{2} & 2 \end{bmatrix}$$

und

$$M_{II} = \begin{bmatrix} 2 & 1 & & & \\ \frac{1}{2} & 2 & \frac{1}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{2} & 2 & \frac{1}{2} \\ & & & 1 & 2 \end{bmatrix}$$

unabhängig von h.

5.2.2 Minimaleigenschaft kubischer Splines

Wir waren bei der Konstruktion der Spline-Funktionen von der Idee ausgegangen, möglichst "glatte" interpolierende Kurven zu finden, wir könnten auch sagen, eine "möglichst wenig gekrümmte". Die *Krümmung* einer parametrisierten Kurve (in der Ebene)

$$y : [t_0, t_n] \to \mathbf{R}, \quad y \in C^2[t_0, t_n]$$

an der Stelle $t \in [t_0, t_n]$ ist gegeben durch

$$\kappa(t) := \frac{y''(t)}{(1 + y'(t)^2)^{3/2}}$$

Der Betrag der Krümmung ist gerade der Reziprokwert $1/r$ des Radius r des Schmiegekreises an die Kurve im Punkt $(t, y(t))$,

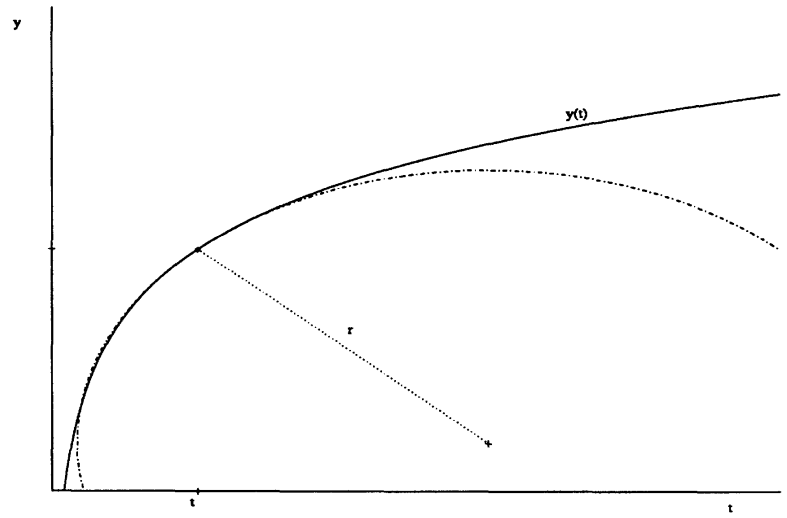


Abbildung 5.12: Schmiegekreis an die Kurve $(t, y(t))$ bei t.

d. h. die Krümmung ist genau dann Null, wenn der Schmiegekreis den Radius ∞ hat, die Kurve also gerade ist.

Zur Vereinfachung betrachten wir anstelle der Krümmung die für kleine $y'(t)$ vernünftige Näherung $y''(t)$

$$\frac{y''(t)}{(1+y'(t)^2)^{3/2}} \to y''(t)$$

und messen die Krümmung der gesamten Kurve, indem wir diese Näherung (zum Quadrat) über das Intervall $[t_0, t_n]$ integrieren, d.h.

$$K(y) := \int_{t_0}^{t_n} y''(t)^2 dt \ .$$

Die interpolierenden kubischen Splines haben nun gerade die Eigenschaft, dieses Funktional zu minimieren.

Satz 5.2.6 *Sei s ein interpolierender kubischer Spline zu den Stützpunkten $(t_0, f_0), \ldots, (t_n, f_n)$ und $y \in C^2[t_0, t_n]$ eine beliebige interpolierende Funktion. Dann gilt*

$$\int_{t_0}^{t_n} s''(t)^2 dt \leq \int_{t_0}^{t_n} y''(t)^2 dt \ . \tag{5.2.7}$$

unter der Bedingung, daß

$$[s(t)''(y(t)' - s(t)')]_{t_0}^{t_n} = 0 \tag{5.2.8}$$

Bemerkung 5.2.7 *Die Bedingung 5.2.8 führt in natürlicher Weise auf die Typen I, II und III.*

Beweis: Mit partieller Integration folgt

$$\int_{t_0}^{t_n} s''(y'' - s'')dt = [s''(y' - s')]_{t_0}^{t_n} - \int_{t_0}^{t_n} s'''(y' - s')dt$$

wobei s''' unstetig in $t_1, \ldots, t_{n-1}$ und konstant

$$s'''(t) = s_i'''(t) = d_i \ \text{ für } \ t \in (t_i, t_{i+1})$$

im Inneren der Teilintervalle (die s_i sind kubische Polynome) und daher

$$\begin{aligned}
\int_{t_0}^{t_n} s''(y'' - s'')dt &= -\sum_{i=1}^{n} \int_{t_{i-1}}^{t_i} d_i(y' - s_i')dt \\
&= -\sum_{i=1}^{n} d_i \int_{t_{i-1}}^{t_i} y' - s_i' dt \\
&= -\sum_{i=1}^{n} d_i [\underbrace{(y(t_i) - s(t_i))}_{=0} - \underbrace{(y(t_{i-1}) - s(t_{i-1}))}_{=0}] \\
&= 0.
\end{aligned}$$

Trivialerweise gilt $y'' = s'' + (y'' - s'')$ und eingesetzt in die rechte Seite von 5.2.7 folgt

$$\begin{aligned}
\int_{t_0}^{t_n} (y'')^2 dt &= \int_{t_0}^{t_n} (s'')^2 dt + \underbrace{2\int_{t_0}^{t_n} s''(y'' - s'')dt}_{=0} + \underbrace{\int_{t_0}^{t_n} (y'' - s'')^2 dt}_{\geq 0} \\
&\geq \int_{t_0}^{t_n} (s'')^2 dt
\end{aligned}$$

□

5.3 Bézier-Technik

Die bisher in diesem Kapitel vorgestellten Themen gehören zum klassischen Teil der numerischen Mathematik, wie schon die auftretenden Namen - wie Lagrange und Newton - verraten. Mit der zunehmenden Bedeutung der Computergraphik sind in jüngerer Zeit (d. h. in den letzten zwanzig Jahren) in der Interpolations- und Approximationstheorie neue Wege beschritten worden, die wir in diesem Abschnitt kurz anreißen wollen. Interessanterweise treffen sich diese neuen Entwicklungen häufig mit recht alten geometrischen Überlegungen, wie sie zum Beispiel schon Möbius durchgeführt hat. Wer sich näher mit diesem Gebiet beschäftigen möchte, dem sei das Skriptum [9] oder das Buch von DE BOOR [18] empfohlen.

Für die folgenden Überlegungen betrachten wir nicht nur reellwertige Polynome, sondern allgemeiner *polynomiale Kurven* in $\mathbf{R}^d$, $d \in \mathbf{N}$.

Definition 5.3.1 *Ein* Polynom *(oder eine* polynomiale Kurve*)* vom Grad n *in* $\mathbf{R}^d$ *ist eine Funktion* P *der Form*

$$P : \mathbf{R} \to \mathbf{R}^d, \quad P(t) = \sum_{i=0}^{n} a_i t^i \quad \textit{mit} \quad a_0, \ldots, a_n \in \mathbf{R}^d, \quad a_n \neq 0$$

Den Raum der Polynome vom Grad kleiner oder gleich n *in* $\mathbf{R}^d$ *bezeichnen wir mit* $\mathbf{P}_d^n$.

Ist $\{P_0, \ldots, P_n\}$ eines Basis von $\mathbf{P}_n$ und $\{e_1, \ldots, e_d\}$ die Standardbasis des $\mathbf{R}^d$, so bilden die Polynome

$$\{e_i P_j \mid \; i = 1, \ldots, d \;\text{ und }\; j = 0, \ldots, n\}$$

eine Basis von $\mathbf{P}_d^n$. Den Graphen $\Gamma_P(t) := (t, P(t))$ eines Polynoms $P \in \mathbf{P}_d^n$ können wir jetzt wieder als ein Polynom $\Gamma_P \in \mathbf{P}_{d+1}^n$ auffassen.

$$\begin{aligned} P(t) &= a_0 + a_1 t + \cdots + a_n t^n \\ \Longrightarrow \Gamma_P(t) &= \binom{0}{a_0} + \binom{1}{a_1} t + \binom{0}{a_2} t^2 + \cdots + \binom{0}{a_n} t^n \end{aligned}$$

Wenn wir ein (gewöhnliches) Polynom $P \in \mathbf{P}_n$ zeichnen, so stellen wir es gerade durch seinen Graphen $\Gamma_P \in \mathbf{P}_2^n$, d.h. eine polynomiale Kurve in der Ebene, dar.

5.3.1 Bernstein-Polynome

Bisher haben wir drei verschiedene Basen des Raumes P_n der Polynome vom Grad kleiner oder gleich n kennengelernt.

- monomiale Basis $1, t, t^2, \ldots, t^n$
- Lagrange-Basis $L_0(t), \ldots, L_n(t)$
- Newton-Basis $w_0(t), \ldots, w_n(t)$

Die beiden letztgenannten Basen sind bereits auf die Interpolation ausgerichtet und hängen von den Knoten $t_0, \ldots, t_n$ ab. Die Basispolynome, die wir jetzt vorstellen wollen, beziehen sich auf ein Intervall $[a, b]$. Sie sind daher hervorragend geeignet für die *lokale* Darstellung eines Polynoms, da die konvexe Hülle der zugehörigen Koeffizienten ein Hüllkörper für die Werte des Polynoms ist. Der erste Schritt besteht in der affinen Transformation auf das Einheitsintervall $I = [0, 1]$

$$\begin{aligned} [a, b] &\rightarrow [0, 1] \\ t &\rightarrow \lambda = \lambda(t) := \frac{t - a}{b - a} \end{aligned} \tag{5.3.1}$$

so daß wir unsere Überlegungen meistens darauf beschränken können. Nach dem binomischen Lehrsatz können wir die Einsfunktion darstellen als

$$1 = ((1 - \lambda) + \lambda)^n = \sum_{i=0}^{n} \binom{n}{i} (1 - \lambda)^{n-i} \lambda^i$$

Die Summanden dieser Zerlegung der Eins sind genau die *Bernstein-Polynome* bzgl. des Intervalls $[0, 1]$. Komponieren wir diese mit obiger affiner Transformation 5.3.1, so erhalten wir die Bernstein-Polynome bzgl. des Intervalls $[a, b]$.

Definition 5.3.2 *(i) Das Polynom* $B_i^n \in \mathbf{P}_n$,

$$B_i^n(\lambda) := \binom{n}{i} (1 - \lambda)^{n-1} \lambda^i, \quad i = 0, \ldots, n$$

heißt i*-tes* Bernstein-Polynom *bzgl.* $[0, 1]$

(ii) Das Polynom $B_i^n[a, b] \in \mathbf{P}_n$

$$B_i^n(t; a, b) := B_i^n(\lambda(t)) = B_i^n(\frac{t - a}{b - a}) = \frac{1}{(b - a)^n} \binom{n}{i} (t - a)^i (b - t)^{n-i}$$

heißt i*-tes* Bernstein-Polynom *bzgl.* $[a, b]$.

Anstelle von $B_i^n(t; a, b)$ schreiben wir im folgenden häufig einfach $B_i^n(t)$, wenn eine Verwechslung mit den Bernstein-Polynomen $B_i^n(\lambda)$ bzgl. $[0, 1]$ ausgeschlossen ist.

In dem folgenden Satz stellen wir die wichtigsten Eigenschaften der Bernstein-Polynome zusammen

Satz 5.3.3 *Für die Bernstein-Polynome* $B_i^n(\lambda)$ *gilt:*

(i) $\lambda = 0$ *ist* i *-fache Nullstelle von* B_i^n.

(ii) $\lambda = 1$ *ist* $(n - i)$*-fache Nullstelle von* B_i^n.

(iii) $B_i^n(\lambda) = B_{n-i}^n(1 - \lambda)$ *für* $i = 0, \ldots, n$ *(Symmetrie).*

(iv) B_i^n *hat im Intervall* $[0, 1]$ *genau ein Maximum, und zwar bei* $\lambda = i/n$.

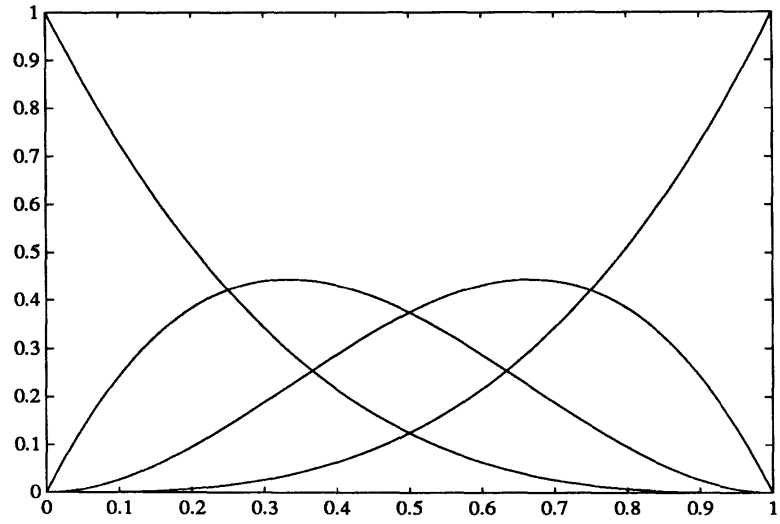


Abbildung 5.13: Bézier–Polynome für $n = 3$.

(v) Die Bernstein-Polynome B_i^n sind nicht negativ auf $[0,1]$ und bilden eine Partition der Eins, d.h.

$$B_i^n(\lambda) \geq 0 \quad \textit{für} \quad \lambda \in [0,1] \quad \textit{und} \quad \sum_{i=0}^{n} B_i^n(\lambda) = 1 \quad \textit{für} \quad \lambda \in \mathbf{R}$$

(vi) Die Bernstein-Polynome bilden eine Basis $\mathcal{B} := \{B_0^n, \ldots, B_n^n\}$ von $\mathbf{P}_n$

Beweis: Die Aussagen (i), (ii), (iii) und (v) sind offensichtlich, (iv) folgt aus der Tatsache, daß

$$\frac{d}{d\lambda} B_i^n(\lambda) = \binom{n}{r} (1-\lambda)^{n-i-1} \lambda^{i-1} (i - n\lambda)$$

für $i = 1, \ldots, n$. Für die letzte Aussage zeigen wir, daß die $n+1$ Polynome B_i^n linear unabhängig sind. Denn falls

$$0 = \sum_{i=0}^{n} b_i B_i^n(\lambda)$$

so gilt wegen (i) und (ii), daß

$$0 = \sum_{i=0}^{n} b_i B_i^n(1) = b_n B_n^n(1) = b_n$$

und daher induktiv $b_0 = \cdots = b_n = 0$. □

Bemerkung 5.3.4 Analoge Aussagen gelten natürlich auch für die Bernstein-Polynome $B_i^n(t; a, b)$. Das Maximum von $B_i^n(t; a, b)$ in $[a, b]$ liegt hier bei

$$t = a + \frac{i}{n}(b - a)$$

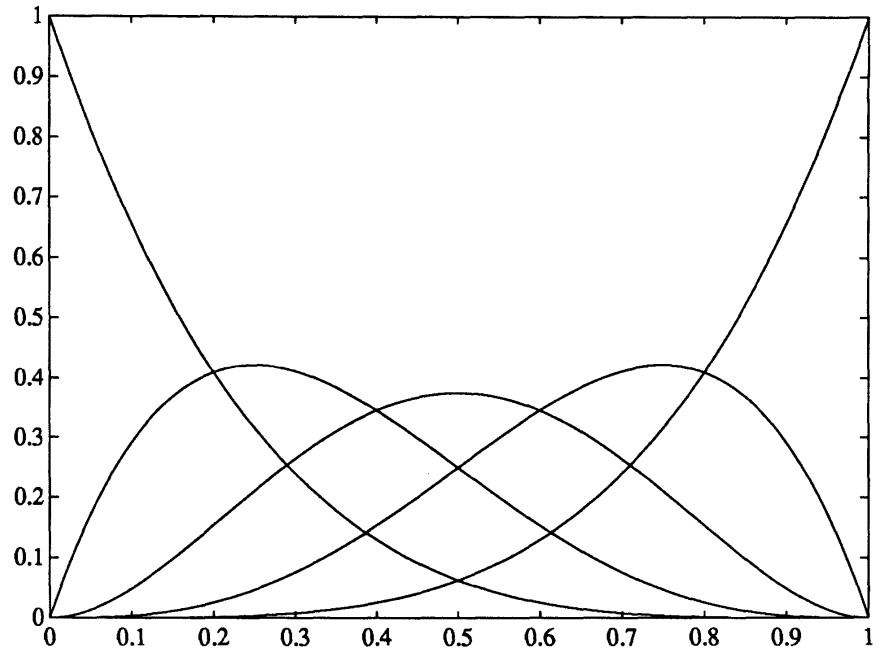


Abbildung 5.14: Bézier–Polynome für $n = 4$.

5.3.2 Bézier–Darstellung von Polynomen

Wir wissen nun, daß wir jedes Polynom $P \in \mathbf{P}_d^n$ bzgl. der Bernstein-Basis schreiben können als Linearkombination

$$P(t) = \sum_{i=0}^{n} b_i B_i^n(t), \quad b_i \in \mathbf{R}^d \tag{5.3.2}$$

Die Koeffizienten $b_0, \ldots, b_n$ heißen *Kontroll-* oder *Bézier-Punkte* von P, der durch sie bestimmte Streckenzug *Bézier-Polygon.* So sind z.B. die Bézier-Punkte von $P(t) = t$ wegen

$$\lambda = \sum_{i=0}^{n} \frac{i}{n} B_i^n(\lambda) \implies t = \sum_{i=0}^{n} (a + \frac{i}{n}(b-a)) B_i^n(t)$$

gerade die Maxima $b_i = a + \frac{i}{n}(b-a)$ der Bernstein-Polynome. Die Bézier-Darstellung des Graphen Γ_P eines Polynoms P wie in 5.3.2 ist daher gerade

$$\Gamma_P(t) = \begin{pmatrix} t \\ P(t) \end{pmatrix} = \sum_{i=0}^{n} \begin{pmatrix} a + \frac{i}{n}(b-a) \\ b_i \end{pmatrix} B_i^n(t)$$

Die Bézier-Punkte des Graphen Γ_P schreiben wir mit einem fettgedruckten $\mathbf{b}$, d.h.

$$\mathbf{b}_i = \begin{pmatrix} a + \frac{i}{n}(b-a) \\ b_i \end{pmatrix}$$

Bemerkung 5.3.5 Die Eigenschaft, daß die Bernstein-Polynome eine Partition der Eins bilden ist äquivalent zu der Tatsache, daß die Bézier-Punkte *affin invariant* sind. Falls $\phi : \mathbf{R}^d \to \mathbf{R}^d$ eine affine Abbildung ist,

$$\begin{aligned} \phi \quad : \quad & \mathbf{R}^d \to \mathbf{R}^d \\ & u \to Au + v, \quad A \in Mat_d(\mathbf{R}), \quad v \in \mathbf{R}^d \end{aligned}$$

so sind die Bilder $\phi(b_i)$ der Bézier-Punkte b_i eines Polynoms $P \in \mathbf{P}_d^n$ die Bézier-Punkte von $\phi \circ P$.

Die Besonderheit der Bernstein-Basis liegt in der nun leicht abzuleitenden geometrischen Bedeutung der Bézier-Punkte. Dazu benötigen wir den Begriff der *konvexen Hülle* einer Menge $A \subset \mathbf{R}^d$, den wir hier kurz wiederholen.

Definition 5.3.6 *(i) Eine Menge $A \subset \mathbf{R}^d$ heißt* konvex, *falls mit je zwei Punkten $x, y \in A$ auch deren Verbindungsstrecke ganz in A liegt, d.h.*

$$[x,y] := \{\lambda x + (1-\lambda)y \mid \lambda \in [0,1]\} \subset A \text{ für alle } x, y \in A$$

(ii) Die konvexe Hülle $\mathrm{co}\,(A)$ *einer Menge $A \subset \mathbf{R}^d$ ist die kleinste konvexe Teilmenge von $\mathbf{R}^d$, die A enthält.*

(iii) Eine Linearkombination der Form

$$x = \sum_{i=1}^{k} \lambda_i x_i, \quad x_i \in \mathbf{R}^d, \quad \lambda_i \geq 0, \quad \sum_{i=1}^{k} \lambda_i = 1$$

heißt Konvexkombination *der $x_1, \ldots, x_k$.*

Bemerkung 5.3.7 *Damit ist sofort klar, daß die konvexe Hülle $\mathrm{co}\,(A)$ von $A \subset \mathbf{R}^d$ die Menge aller Konvexkombinationen von Punkten aus A ist, d. h.*

$$\begin{aligned} \mathrm{co}\,(A) &= \bigcap \{B \subset \mathbf{R}^d | B \text{ konvex mit } A \subset B\} \\ &= \left\{ x = \sum_{i=1}^{m} \lambda_i x_i \mid m \in \mathbf{N}, \quad x_i \in A, \quad \lambda_i \geq 0, \quad \sum_{i=1}^{m} \lambda_i = 1 \right\} \end{aligned}$$

Satz 5.3.8 *Das Bild $P([a,b])$ eines Polynoms $P \in \mathbf{P}_d^n$ liegt in der konvexen Hülle der Bézier-Punkte b_i von P, d.h.*

$$P(t) \in \mathrm{co}\,(b_0, \ldots, b_n) \text{ für } t \in [a,b]$$

Insbesondere liegt der Graph des Polynoms für $t \in [a,b]$ in der konvexen Hülle der Punkte $\mathbf{b}_i$.

Beweis: Da $B_i^n(t) \geq 0$ für $t \in [a,b]$ und $\sum_{i=0}^{n} B_i^n(t) = 1$, ist

$$P(t) = \sum_{i=0}^{n} b_i B_i^n(t),$$

eine Konvexkombination der Bézier-Punkte $b_0, \ldots, b_n$. Die zweite Aussage folgt aus der Bézier-Drastellung (5...) des Graphen Γ_P von P. □

Anschaulich bedeutet dies für ein (gewöhnliches) kubisches Polynom $P \in \mathbf{P}_3$, daß der Graph von P für $t \in [a,b]$ ganz in der konvexen Hülle der vier Bézier-Punkte $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ und $\mathbf{b}_4$ liegt. Der Name Kontrollpunkt erklärt sich nun dadurch, daß sich die Punkte $\mathbf{b}_i$ aufgrund ihrer geometrischen Bedeutung gut zur Steuerung einer polynomialen Kurve eignen. Der Kontrollpunkt b_i hat wegen Satz 5.3.3 gerade an der Stelle $\lambda = i/n$ das größte "Gewicht" $B_i^n(\lambda)$, über der er aufgetragen wird. Dies ist ein Grund dafür, daß die From der Kurve zwischen a und b eng mit dem Bézier-Polygon zusammenhängt, wie dies die Zeichnung andeutet. Um dies weiter zu analysieren, berechnen wir die Ableitungen eines Polynoms in Bézier-Darstellung.

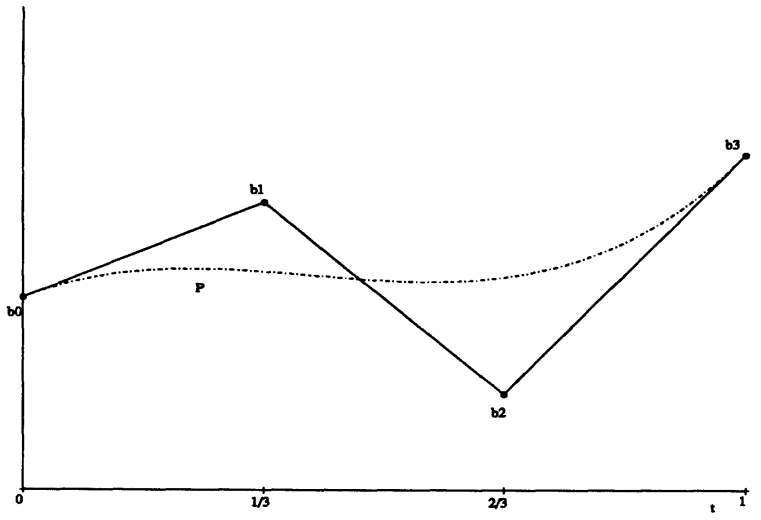


Abbildung 5.15: Ein kubisches Polynom mit seinem Bézier–Polygon.

Lemma 5.3.9 *Für die Ableitung der Bernstein-Polynome $B_i^n(t)$ bzgl. $[a,b]$ gilt*

$$\frac{d}{dt}B_i^n(t) = \begin{cases} -\frac{n}{b-a}B_0^{n-1}(t) & \text{für} \quad i=0 \\ \frac{n}{b-a}(B_{i-1}^{n-1}(t) - B_i^{n-1}(t)) & \text{für} \quad i=1,\ldots,n-1 \\ \frac{n}{b-a}B_{n-1}^{n-1}(t) & \text{für} \quad i=n \end{cases}$$

Beweis: Es ist

$$\frac{d}{d\lambda}B_i^n(\lambda) = \binom{n}{i}[i(1-\lambda)^{n-i}\lambda^{i-1} - (n-i)(1-\lambda)^{n-i-1}\lambda^i]$$

$$= \begin{cases} -n(1-\lambda)^{n-1} = -nB_0^{n-1} & \text{für} \quad i=0 \\ n(B_{i-1}^{n-1}(\lambda) - B_i^{n-1}(\lambda)) & \text{für} \quad i=1,\ldots,n-1 \\ nB_{n-1}^{n-1}(t) & \text{für} \quad i=n \end{cases}$$

und trivialerweise

$$\frac{d}{dt}\lambda(t) = \frac{1}{b-a}.$$

□

Satz 5.3.10 *Sei $P(t) = \sum_{i=0}^n b_i B_i^n(t)$ ein Polynom in Bézier-Darstellung bzgl. $[a,b]$. Dann gilt für die k-te Ableitung von P*

$$P^{(k)}(t) = \frac{1}{(b-a)^k}\frac{n!}{(n-k)!}\sum_{i=0}^{n-k}\Delta^k b_i B_i^{n-k}(t),$$

wobei Δ der Vorwärtsdifferenzenoperator

$$\begin{aligned} \Delta^1 b_i &:= b_{i+1} - b_i \\ \Delta^k b_i &:= \Delta^{k-1}b_{i+1} - \Delta^{k-1}b_i \end{aligned}$$

Beweis: Induktion über k, siehe Aufgabe 5.1. □

Korollar 5.3.11 *Insbesondere erhält man für die Randpunkte a, b die Werte*

$$\begin{aligned} P^{(k)}(a) &= \frac{1}{(b-a)^k}\frac{n!}{(n-k)!}\Delta^k b_0, \quad da\ B_i^{n-k}(a) = \delta_{0,i} \\ P^{(k)}(b) &= \frac{1}{(b-a)^k}\frac{n!}{(n-k)!}\Delta^k b_{n-k}, \quad da\ B_i^{n-k}(b) = \delta_{n-k,i} \end{aligned}$$

also speziell bis zur zweiten Ableitung

a)	$P(a) = b_0$	$P(b) = b_n$
b)	$P'(a) = \frac{n}{b-a}(b_1 - b_0)$	$P'(b) = \frac{n}{b-a}(b_n - b_{n-1})$
c)	$P''(a) = \frac{n(n-1)}{(b-a)^2}(b_2 - 2b_1 + b_0)$	$P''(b) = \frac{n(n-1)}{(b-a)^2}(b_n - 2b_{n-1} + b_{n-2})$

Die Kurve P verläuft also durch Anfangs- und Endpunkt des Bézier-Polygons und die Tangenten der Kurve in den Randpunkten a, b sind gerade die Anfangs- bzw. Endseite des Polygons.

Beispiel 5.3.12 Als Beispiel betrachten wir noch einmal das kubische Hermite-Polynom H_0^3 mit

$$H_0^3(0) = 1, \frac{d}{dt}H_0^3(0) = 0, H_0^3(1) = 0, \frac{d}{dt}H_0^3(1) = 0$$

Wegen der Eigenschaften a) und b) von Korollar 5.3.11 können wir das zugehörige Bézier-Polygon sofort zeichnen:

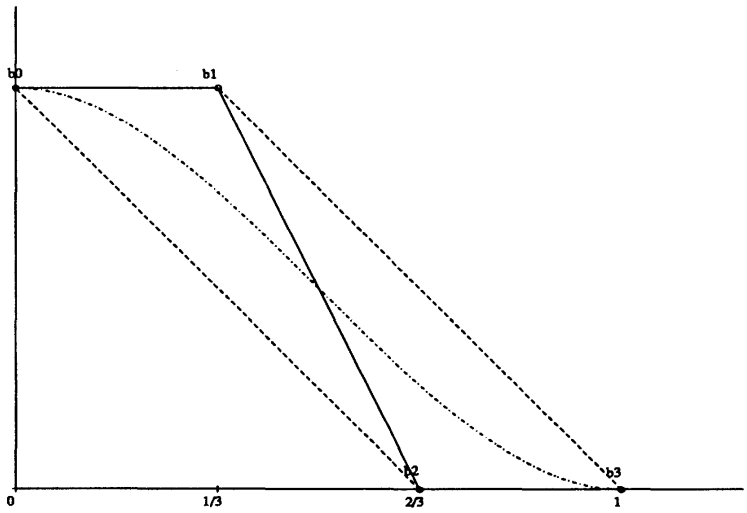


Abbildung 5.16: Das Hermite–Polynom H_0^3 mit seinem Bézier–Polygon.

Zusammen mit der Eigenschaft, daß H_0^3 für $t \in [0,1]$ in der konvexen Hülle der Punkte $b_0, \ldots, b_3$ liegen muß, gibt uns dies bereits eine recht genaue Vorstellung von H_0^3 und wir können die Bézier-Darstellung von H_0^3 direkt angeben (siehe auch Aufgabe 5.3)

$$H_0^3 = B_0^3 + B_1^3$$

5.3.3 Algorithmus von de Casteljau

Die Bedeutung der Bézier-Darstellung beruht neben der geometrischen Interpretation der Bézier-Punkte vor allem darauf, daß es ähnlich wie beim Schema von Aitken und Neville einen auf fortgesetzter Konvexkombination beruhenden Algorithmus gibt, der neben dem Funktionswert $P(t)$ an einer beliebigen Stelle t auch noch Informationen über die Ableitungen liefert. Darüber hinaus kann derselbe Algorithmus benutzt werden, um die Bézier-Kurve in zwei Teilsegmente zu unterteilen. Wiederholt man diese Unterteilung in Teilsegmente, so konvergiert die Folge der Bézier-Polygone extrem schnell (bei Halbierung des Intervalls exponentiell) gegen die Kurve, so daß sich dieses Verfahren sehr gut eignet, um eine Kurve z. B. in der Computergraphik effektiv zu zeichnen. (Dieses Konstruktionsprinzip ist auch maßgeschneidert für die Steuerung einer Fräse, die ja nur Material "wegnehmen" kann.)

Lemma 5.3.13 *Für die Bernstein-Polynome $B_i^n(\lambda)$ gilt die Rekursionsformel*

$$B_i^n(\lambda) = \lambda B_{i-1}^{n-1}(\lambda) + (1-\lambda) B_i^{n-1}(\lambda) \quad \textit{für} \quad i = 1, \ldots, n \textit{ und } \lambda \in \mathbf{R}$$

Beweis: Die Behauptung folgt aus der Definition und der Tatsache, daß

$$\binom{n}{i} = \binom{n-1}{i-1} + \binom{n-1}{i}$$

□

Den *Algorithmus von de Casteljau* formulieren wir in dem folgenden Satz.

Satz 5.3.14 *Sei $P(t) = \sum_{i=0}^n b_i B_i^n(t)$ ein Polynom in Bézier-Darstellung bzgl. $[a,b]$ und b_i^k für $k = 0, \ldots, n$ und $i = 0, \ldots, n-k$ rekursiv erklärt durch*

a) $b_i^0 := b_i$ *für* $i = 0, \ldots, n$

b) $b_i^k := (1-\lambda) b_i^{k-1} + \lambda b_{i+1}^{k-1}$ *für* $k = 1, \ldots, n$ *und* $i = 0, \ldots, n-k$

mit $\lambda := (t-a)/(b-a)$ für ein festes $t \in \mathbf{R}$. Dann gilt

$$P^{(k)}(t) = \frac{1}{(b-a)^k} \frac{n!}{(n-k)!} \Delta^k b_0^{n-k} \quad \textit{für} \quad k = 0, \ldots, n,$$

wobei Δ auf dem unteren Index operiert, d.h.

$$\Delta b_i^k = b_{i+1}^k - b_i^k$$

Bemerkung 5.3.15 Die b_i^k können in dem *Schema von de Casteljau* angeordnet werden:

$$
\begin{array}{lccccccc}
b_1 = b_n^0 & & & & & & & \\
& \searrow & & & & & & \\
b_1 = b_{n-1}^0 & \to & b_{n-1}^1 & & & & & \\
& \searrow & & \searrow & & & & \\
b_1 = b_{n-2}^0 & \to & b_{n-2}^1 & \to & b_{n-2}^2 & & & \\
\vdots & & & & \vdots & & & \\
\vdots & & & & \vdots & & & \\
b_1 = b_1^0 & \to & \cdots & \to & b_1^{n-1} & & & \\
& \searrow & & & & \searrow & & \\
b_1 = b_0^0 & \to & \cdots & \to & b_0^{n-1} & \to & b_0^n &
\end{array}
\tag{5.3.3}
$$

Die k-te Ableitung $P^{(k)}(t)$ an der Stelle t berechnet sich aus der $(n-k)$-ten Spalte. Insbesondere gilt

$$
\begin{aligned}
&\text{a)} \quad P(t) = b_0^n \\
&\text{b)} \quad P'(t) = \frac{n}{b-a}(b_1^{n-1} - b_0^{n-1}) \\
&\text{c)} \quad P''(t) = \frac{n(n-1)}{(b-a)^2}(b_2^{n-2} - 2b_1^{n-2} + b_0^{n-2})
\end{aligned}
\tag{5.3.4}
$$

Beweis: (von Satz 5.3.14 Wir zeigen, daß

$$
b_i^k = \sum_{j=i}^{k+i} b_j B_{j-i}^k(t) \tag{5.3.5}
$$

Die Behauptung folgt dann aus Satz 5.3.14, da dann

$$
\begin{aligned}
P^{(k)}(t) &= \frac{1}{(b-a)^k}\frac{n!}{(n-k)!}\sum_{i=0}^{n-k}\Delta^k b_i B_i^{n-k}(t) \\
&= \frac{1}{(b-a)^k}\frac{n!}{(n-k)!}\Delta^k\sum_{i=0}^{n-k} b_i B_i^{n-k}(t) \\
&= \frac{1}{(b-a)^k}\frac{n!}{(n-k)!}\Delta^k b_0^{n-k}
\end{aligned}
$$

Für $k = 0$ ist 5.3.5 richtig, da

$$
b_i^0 = b_i = b_i B_0^0 = \sum_{j=i}^{i} b_j B_{j-i}^0
$$

Sei nun induktiv vorausgesetzt, daß

$$
b_i^{k-1} = \sum_{j=i}^{k-1+i} b_j B_{j-i}^{k-1}
$$

$$b_{i+1}^{k-1} = \sum_{j=i+1}^{(k-1)+(i+1)} b_j b_{j-i-1}^{k-1} = \sum_{j=i+1}^{k+i} b_j B_{j-i-1}^{k-1}$$

Dann folgt mit Lemma 5.3.13

$$\begin{aligned} b_i^k &= (1-\lambda)b_i^{k-1} + \lambda b_{i+1}^{k-1} \\ &= (1-\lambda)\sum_{j=i}^{k+i-1} b_j B_{j-1}^{k-1} + \lambda \sum_{j=i+1}^{k+i} b_j B_{j-i-1}^{k-1} \\ &= \sum_{j=i+1}^{k+i-1} b_j[(1-\lambda)B_{j-i}^{k-1} + \lambda B_{j-i-1}^{k-1}] + \underbrace{(1-\lambda)b_i B_0^{k-1}}_{=b_i B_0^k} + \underbrace{\lambda b_{k+i} B_{k-1}^{k-1}}_{=b_{k+i}B_k^k} \\ &= \sum_{j=i}^{k+i} b_j B_{j-i}^k \end{aligned}$$

□

Wie wir oben bereits erwähnt haben, kann man den Algorithmus von de Casteljau auch benutzen, um eine Bézier-Kurve in zwei Teilsegmente zu zerlegen.

Satz 5.3.16 *Sei* $P(t) = \sum_{i=0}^n b_i B_i^n(t;a,c)$ *eine Bézier-Kurve bzgl.* $[a,c]$ *und* $a \le b \le c$. *Dann ergeben sich die Bézier-Punkte der Teilkurven über* $[a,b]$ *bzw.* $[b,c]$ *aus der unteren Zeile bzw. der Diagonale des de Casteljau-Schemas* $b_i^k = b_i^k(b)$, *d.h. es gilt*

$$P(t) = \sum_{i=0}^n b_i B_i^n(t;a,c) = \begin{cases} \sum_{i=0}^n b_0^i B_i^n(t;a,b) & \text{für} \quad a \le t \le b \\ \sum_{i=0}^n b_i^{n-i} B_i^n(t;b,c) & \text{für} \quad b \le t \le c \end{cases}$$

Beweis: siehe z. B [17] □

Wiederholt man diese Unterteilung, so konvergieren die zugehörigen Bézier-Polygone gegen die Kurve. Daher bietet die fortgesetzte Unterteilung zusammen mit dem Algorithmus von de Casteljau ein effektives Verfahren zum Zeichnen polynomialer Kurven, welches nur auf konvexer Kombination der Ausgangskoeffizienten beruht und deswegen auch äußerst stabil ist.

Beispiel 5.3.17 Fortgesetzte Unterteilung einer kubischen Bézier-Kurve (jeweils Halbierung des Intervalls)

5.3.4 Bézier-Darstellung von Splines

Nachdem es sich als günstig erwiesen hat, Polynome als Linearkombination der Bernstein-Polynome zu schreiben, liegt es nahe, eine ähnliche Darstellung für Splines zu suchen. Wir beschränken uns wieder auf kubische Splines und der Einfachheit halber auf äquidistante Stützstellen

$$t_i = i \quad \text{für} \quad i = 0, \ldots, n$$

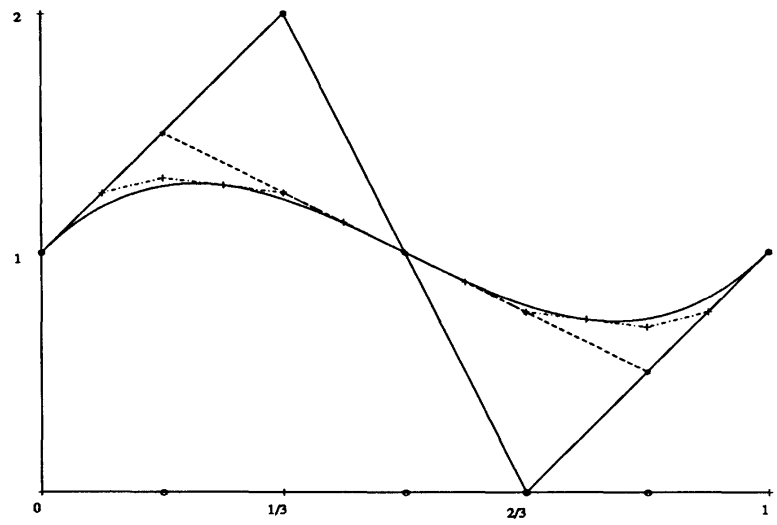


Abbildung 5.17: Fortgesetzte Unterteilung einer kubischen Bézier-Kurve.

Jedes Teilpolynom $s_i \in \mathbf{P}_3$ besitzt eine Bézier-Darstellung

$$\begin{aligned} s_i(t) &= \sum_{j=0}^{3} b_j^{(i)} B_j^3(t; t_i, t_{i+1}) \\ &= \sum_{j=0}^{3} b_j^{(i)} \binom{3}{j} (1-\lambda_i)^{n-j} \lambda_i^j \quad \text{mit} \quad \lambda_i := \frac{t-t_i}{t_{i+1}-t_i} \end{aligned}$$

bzgl. des Stützintervalls $[t_i, t_{i+1}]$. Aufgrund der Verheftungsbedingungen 5.2.3 a) gilt

$$b_3^{i-1)} = s_{i-1}(t_i) = s_i(t_i) = b_0^{(i)} \quad \text{für} \quad i = 1, \ldots, n-1,$$

so daß wir nur drei Bézier-Punkte pro Teilpolynom s_i für $i = 1, \ldots, n-1$ benötigen. Um die Notation etwas zu erleichtern, setzen wir für $i = 0, \ldots, n-1$

$$\begin{aligned} b_{3i} &:= b_0^{(i)} \\ b_{3i+1} &:= b_1^{(i)} \\ b_{3i+2} &:= b_2^{(i)} \\ b_{3n} &:= b_3^{(n-1)} \end{aligned}$$

so daß

$$s_i(t) = \sum_{j=0}^{3} b_{3i+j} B_j^3(t; t_i, t_{i+1}) \quad \text{für} \quad i = 0, \ldots, n-1 \, .$$

Die ersten Ableitungen von s_{i-1} und s_i stimmen nach Korollar cor:bezier-ableitungen in t_i überein, falls

$$s'_{i-1}(t_i) = 3(b_{3i} - b_{3i-1}) = 3(b_{3i+1} - b_{3i}) = s'_i(t_i) \quad \text{für} \quad i = 1, \ldots, n-1$$

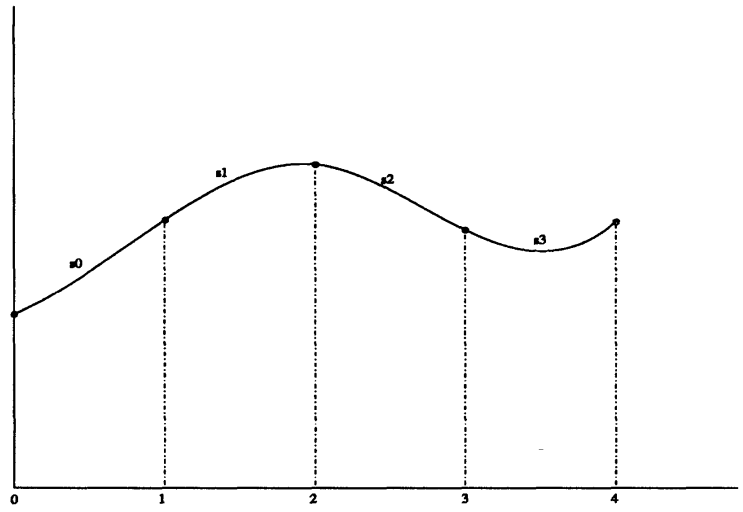


Abbildung 5.18: Kubischer Spline mit äquidistanten Stützstellen.

d.h.

$$b_{3i} = \frac{b_{3i-1} + b_{3i+1}}{2} \quad \text{für} \quad i = 1, \ldots, n-1 \tag{5.3.6}$$

Anschaulich gesprochen müssen die Bézier-Punkte $\mathbf{b}_{3i}$ die Strecke zwischen $\mathbf{b}_{3i-1}$ und $\mathbf{b}_{3i+1}$ halbieren. Analog ist die zweimalige Differenzierbarkeit von s in t_i äquivalent zu

$$s''_{i-1}(t_i) = 6(b_{3i-2} - 2b_{3i-1} + b_{3i}) = 6(b_{3i} - 2b_{3i+1} + b_{3i+2}) = s''_i(t)$$

d.h.

$$2b_{3i-1} - b_{3i-2} = 2b_{3i+1} - b_{3i+2} =: d_i \quad \text{für} \quad i = 1, \ldots, n-1 \tag{5.3.7}$$

Hier müssen die Punkte $b_{3i\pm1}$ die Strecke zwischen dem Hilfspunkt d_i und $b_{3i\pm2}$ halbieren. Gegenüber der Spline-Darstellung im Abschnitt 5.2.1 zeigt sich wiederum, daß die Bézier-Darstellung stets einfach geometrisch zu interpretieren ist. Die Bézier-Punkte $b_{3i\pm1}$ können wir wegen 5.3.6 und 5.3.7 aus den Hilfspunkten d_i berechnen:

$$\begin{aligned} 3b_{3i-1} &= d_{i-1} + 2d_i \\ 3b_{3i+1} &= 2d_i + d_{i+1} \end{aligned} \tag{5.3.8}$$

Die Punkte b_{3i+1} und b_{3i+2} dreiteilen gerade die Strecke zwischen d_i und d_{i+1} für $i = 0, \ldots, n-1$. Setzen wir die Gleichungen 5.3.8 in 5.3.6 ein, so folgt

$$d_{i-1} + 4d_i + d_{i+1} = 6b_{3i} \quad \text{für} \quad i = 1, \ldots, n-1$$

Zusammen mit 5.3.8 für $i = n$ bzw. $i = 0$ lassen sich die $d_0, \ldots, d_n$ durch das lineare

Gleichungssystem

$$\begin{bmatrix} 2 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 2 \end{bmatrix} \begin{pmatrix} d_0 \\ \vdots \\ \vdots \\ \vdots \\ d_n \end{pmatrix} = \begin{pmatrix} 3b_1 \\ 6b_3 \\ \vdots \\ 6b_{3n-3} \\ 3b_{3n-1} \end{pmatrix}$$

aus den Werten b_1, b_{3i}, b_{3n-1} berechnen.

Für die Typen I und II erhalten wir zusätzlich zur Interpolationseigenschaft

$$b_{3i} = s(t_i) = f_i \text{ für } i = 0, \ldots, n$$

die Gleichungen

$$\begin{aligned} \textbf{Typ I} \quad : \quad s''(0) &= 0 \implies d_0 = b_0 \\ s''(n) &= 0 \implies d_n = b_n \end{aligned}$$

$$\begin{aligned} \textbf{Typ II} \quad : \quad s'(0) &= f_0' = 3(b_1 - b_0) \implies b_1 = f_0'/3 + b_0 \\ s'(n) &= f_n' = 3(b_{3n} - b_{3n-1}) \implies b_{3n-1} = b_{3n} - f_n'/3 \end{aligned}$$

Tatsächlich verwendet man heute häufig schon nicht mehr die Bézier-Darstellung von Splines, sondern die sogenannten *B-Splines* $N_k(t)$. In unserem Spezialfall der *kubischen B-Splines* sind dies gerade die Splines mit

$$d_i = \delta_{ik}$$

deren Bézier-Punkte wir mit den Formeln 5.3.6 und 5.3.8 leicht berechnen können

j	$\ldots$	-4	-3	-2	-1	0	1	2	2	4	$\ldots$
b_{3k+j}	0	0	$\frac{1}{6}$	$\frac{2}{6}$	$\frac{4}{6}$	$\frac{4}{6}$	$\frac{4}{6}$	$\frac{2}{6}$	$\frac{1}{6}$	0	0

Das "B" in der Bezeichnung "B-Spline" steht für Basis und das natürlich nicht ohne Grund. Die $n-3$ B-Splines $N_{-1}(t), \ldots, N_{n+1}(t)$ bilden eine Basis der kubischen Splines über den Stützpunkten $0, \ldots, n$. Die folgenden Eigenschaften erinnern uns an die der Bernstein-Polynome und lassen den (begründeten) Verdacht aufkommen, daß sich dies ähnlich positiv auf die Darstellung eines Splines als Linearkombination der B-Splines auswirkt wie bei der Bézier-Darstellung eines Polynoms.

Bemerkung 5.3.18 *Die B-Splines $N_k(t)$ haben folgende Eigenschaften:*

a) $N_k(t) \geq 0$ *für alle* $t \in \mathbf{R}$.

b) $N_k(t) = 0$ *für alle* t *mit* $t \leq k-2$ *oder* $t \geq k+2$, *d. h.* $\operatorname{supp} N_k(t) \subset [k-2, k+2]$.

c) $\sum_{k=-1}^{n-1} N_k(t) = 1$ *für* $0 \leq t \leq n$

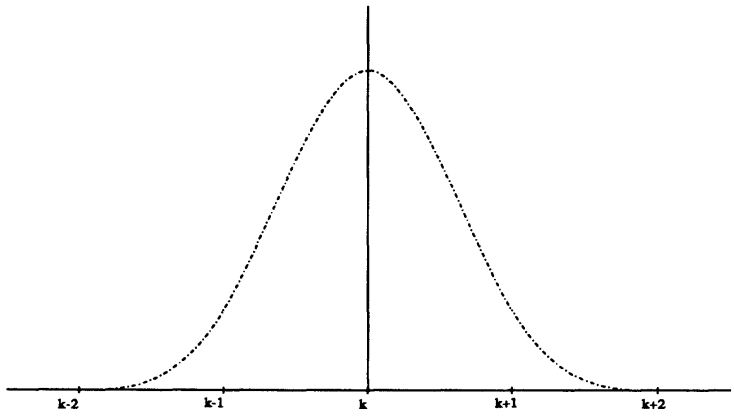


Abbildung 5.19: Kubischer B-Spline $N_k(t)$.

5.4 Übungen

Aufgabe 5.1 Zeigen Sie, daß die Ableitungen eines Polynoms in Bézier-Darstellung bzgl. des Intervalls $[t_0, t_1]$

$$P(t) = \sum_{i=0}^{n} b_i B_i^n(\lambda), \quad \lambda := \frac{t - t_0}{t_1 - t_0},$$

gegeben sind durch

$$\frac{d^k}{dt^k} P(t) = \frac{n!}{(n-k)! h^k} \sum_{i=0}^{n-k} \Delta^k b_i B_i^{n-k}(\lambda), \quad h := t_1 - t_0.$$

Aufgabe 5.2 Wir haben für den Raum $\mathbf{P}_3$ der Polynome vom Grad kleiner oder gleich 3 die monomiale Basis $\{1, t, t^2, t^3\}$, die Bernstein-Basis $\{B_0^3(t), B_1^3(t), B_2^3(t), B_3^3(t)\}$ bzgl. des Intervalls $[0, 1]$ und die Hermite-Basis $\{H_0^3(t), H_1^3(t), H_2^3(t), H_3^3(t)\}$ für die Knoten t_0, t_1 kennengelernt. Bestimmen Sie die Matrizen der Basiswechsel.

Aufgabe 5.3 Geben Sie die Bézier-Darstellung bzgl. $[0, 1]$ der Hermite-Polynome H_i^3 für die Knoten t_0, t_1 an und skizzieren Sie die Hermite-Polynome zusammen mit den Bézier-Polygonen.

6 Numerische Quadratur

Ein relativ häufiges Problem ist die Berechnung des Riemann-Integrals

$$I(f) := I_a^b(f) := \int_a^b f(t)\, dt$$

wobei formal f eine stückweise stetige Funktion auf dem Intervall $[a, b]$, jedoch in den Anwendungen i. a. stückweise glatt ist. (Eine stückweise stetige und nicht glatte Funktion ist auf einem Rechner nicht implementierbar.)

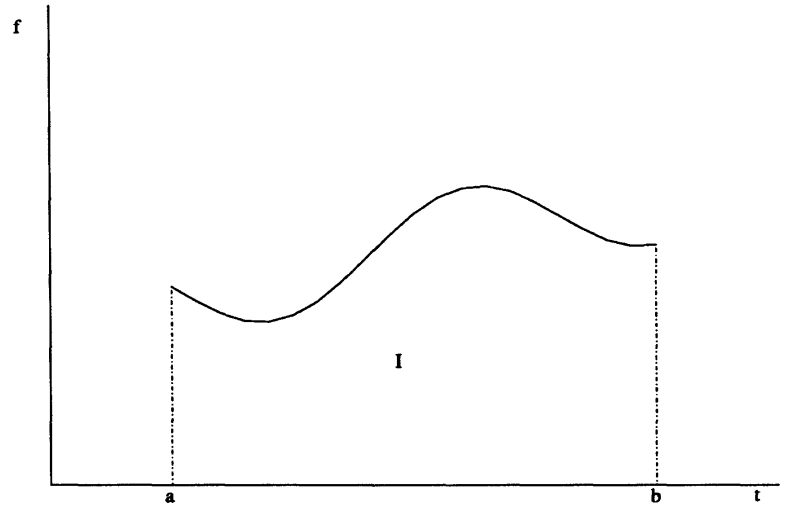


Abbildung 6.20: Aufgabe der Quadratur.

In der Analysis lernt man zahlreiche Techniken kennen, ein derartiges Integral zu "lösen" in dem Sinne, daß ein "einfacherer" Ausdruck für $I_a^b(f)$ gefunden wird. Ist dies möglich, so sagt man, das Integral sei *analytisch* oder *in geschlossener Form* lösbar. Diese beiden Begriffe besagen jedoch nicht viel mehr, als daß die "analytischen Ausdrücke" mathematisch besser bekannt und deswegen dem ursprünglichen Integralausdruck vorzuziehen sind. Ist man jedoch an dem konkreten Zahlenwert $I_a^b(f)$ interessiert, so ist häufig günstiger, dieses Problem direkt anzugehen (vgl. 3.3), auch wenn eine Lösung in geschlossener Form vorliegt. Als Beispiel betrachte man folgendes Integral, bei dem die Auswertung des analytischen Ausdrucks gewiß abschreckend ist.

Beispiel 6.0.1 *aus* GRÖBNER-HOFREITER, *Integraltafeln 1. Teil*

$$\int \frac{x^k}{(ax+b)^m (cx+d)^n}\, dx$$

$$= A_0 \log(ax+b) + B_0 \log(cx+d) - \sum_{\mu=1}^{m-1} \frac{A_\mu}{\mu(ax+b)^\mu} - \sum_{\nu=1}^{n-1} \frac{B_\nu}{\nu(ax+b)^\nu} + C$$

für $k = 0, 1, \ldots, m+n-1$ *mit*

$$A_\mu = (-1)^{m+k+\mu+1} a^{n-k-1} \sum_{j=0}^{m-\mu-1} \binom{k}{j} \binom{m+n-\mu-j-2}{n-1} \frac{b^{k-j} c^{m-\mu-j-1}}{(ad-bc)^{m+n-\mu-j-1}}$$

für $\mu = 0, 1, \ldots, m-1$ *und*

$$B_\nu = (-1)^m c^{m-k-1} \sum_{j=0}^{n-\nu-1} \binom{k}{j} \binom{m+n-\nu-j-2}{m-1} \frac{(-d)^{k-j} a^{n-\nu-j-1}}{(ad-bc)^{m+n-\nu-j-1}}$$

für $\nu = 0, 1, \ldots, n-1$. □

Die direkte (algorithmische) Berechnung von $I_a^b(f)$ bezeichnet man als *numerische Quadratur.* Dieser Begriff läßt uns unwillkürlich an die *Quadratur des Kreises* denken, das wegen der Transzendenz von π nicht lösbare Problem, mit Zirkel und Lineal ein Quadrat mit dem Flächeninhalt des Einheitskreises zu konstruieren. Die heutige Bedeutung von Quadratur ist aus dieser "Konstruktion eines Quadrates gleichen Flächeninhalts" erwachsen.

Häufig findet man auch den Begriff Integration, der auch die Lösung von Differentialgleichungen beschreibt. Tatsächlich gilt ja, daß die Berechnung des Integrals $I_a^b(f)$ formal dem Lösen der *Anfangswertaufgabe*

$$y'(t) = f(t), \quad y(a) = 0 , \quad t \in [a, b]$$

entspricht, da für die Lösungsfunktion $y(t)$

$$y(b) = I_a^b(f)$$

gilt. Wir werden auf diese Formulierung noch im Verlauf des Kapitels zurückkommen.

Aus der Analysis kennen wir unter anderem folgende Eigenschaften des Riemann-Integrals.

Bemerkung 6.0.2 *Für stückweise stetige Funktionen* f, f_1, f_2 *auf* $[a, b]$ *gilt:*

(I) $I_a^b(c_1 f_1 + c_2 f_2) = c_1 I_a^b(f_1) + c_2 I_a^b(f_2)$ *für* $c_1, c_2 \in \mathbf{R}$ *(*I_a^b *ist ein* lineares Funktional*)*

(II) $I_a^\tau + I_\tau^b = I_a^b$ *für* $\tau \in [a, b]$ *(*I *ist* additiv*)*

(III) Falls f *nicht negativ* $(f \geq 0)$ *und nicht die Nullfunktion* $(f \neq 0)$ *ist, d.h.*

$$\begin{aligned} f(t) &\geq 0 \quad \text{für alle} \quad t \in [a, b] \\ f(t_0) &\neq 0 \quad \text{für ein} \quad t_0 \in [a, b] \end{aligned}$$

so ist das Integral (strikt) positiv, d.h.

$$I_a^b(f) > 0$$

Wie immer erwarten wir von einem Verfahren zur Berechnung des Integrals, daß es derartige wichtige Eigenschaften erhält. Es wird sich leider zeigen, daß sich nur die erste Eigenschaft vollständig retten läßt.

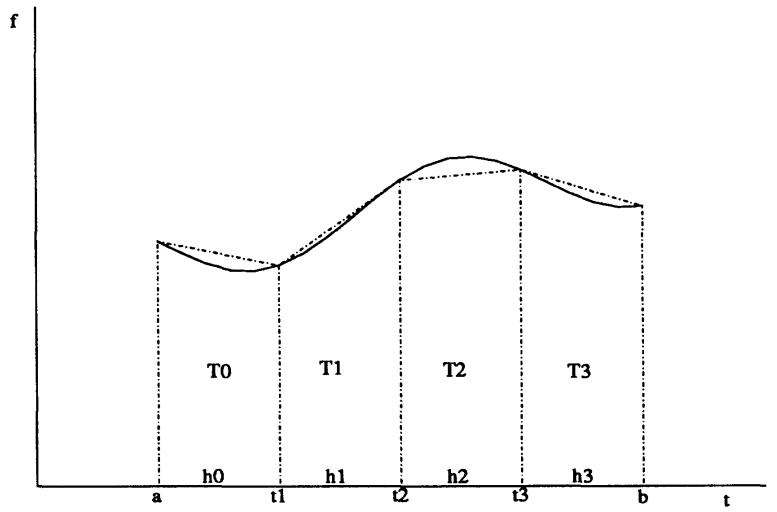


Abbildung 6.21: Trapezsumme mit äquidistanten Stützstellen

6.1 Newton-Cotes-Formeln

Eine erste einfache Methode zur Berechnung von $I_a^b(f)$ ist die sogenannte *Trapezsumme.* Wir zerlegen das Intervall in n Teilintervalle $[t_i, t_{i+1}]$ $(i = 0, \ldots, n-1)$ der Länge $h_i := t_{i+1} - t_i$

$$a = t_0 < t_1 < \cdots < t_n = b$$

und approximieren das Integral $I_a^b(f)$ durch die Summe der Trapezflächen

$$T^{(n)} := \sum_{i=0}^{n-1} T_i, \quad T_i := \frac{h_i}{2}(f(t_i) + f(t_{i+1}))$$

Vergleichen wir die Trapezsumme $T^{(n)}$ mit den Riemann'schen Unter- bzw. Obersummen

$$\begin{aligned} R_U^{(n)} &:= \textstyle\sum_{i=0}^{n-1} h_i \min_{t\in[t_i,t_{i+1}]} f(t) \\ R_O^{(n)} &:= \textstyle\sum_{i=0}^{n-1} h_i \max_{t\in[t_i,t_{i+1}]} f(t) \end{aligned}$$

so ist offensichtlich, daß

$$R_U^{(n)} \leq T^{(n)} \leq R_0^{(n)}$$

Für stetiges $f \in C^0[a,b]$ folgt daher aus der Konvergenz der Riemann'schen Summen auch die der Trapezsumme:

$$\begin{array}{ccccc} R_U^{(n)} & \leq & T^{(n)} & \leq & R_O^{(n)} \\ \downarrow & & \downarrow & & \downarrow \qquad \text{für } n \to \infty, \ h_i \leq h \to 0 \\ I_a^b(f) & & I_a^b(f) & & I_a^b(f) \end{array}$$

Weiter sehen wir, daß $T^{(n)}$ ein lineares Funktional definiert, die Eigenschaft (II) jedoch nur erfüllt, falls $\tau \in \{t_0, \ldots, t_n\}$ und (III) nur gilt, wenn $f(t_i) > 0$ für ein $t_i \in \{t_0, \ldots, t_n\}$.

Die Idee bei der Trapezsumme bestand darin, die Funktion f durch eine Approximation $\hat{f}$ zu ersetzen, für die sich die Quadratur einfach ausführen läßt, und $I(\hat{f})$ als Approximation von $I(f)$ anzusehen, also

$$\hat{I}(f) := I(\hat{f})$$

Dabei können natürlich nicht nur lineare (wie bei der Trapezregel), sondern beliebige Approximationen, wie sie z.B. im letzten Kapitel vorgestellt wurden, benutzt werden. Insbesondere erhält man für die Polynom-Interpolation an den Stützstellen $(t_i, f(t_i))$, $i = 0, \ldots, n$

$$\hat{f}(t) = \sum_{i=0}^{n} f(t_i) L_{i,n}(t),$$

wobei $L_{i,n}(t)$ das i-te Lagrange-Polynom zu den Stützstellen $t_0, \ldots, t_n$ ist. Dieser Ansatz liefert die Quadraturformel

$$\hat{I}(f) := I(\hat{f}) = \sum_{i=0}^{n} \alpha_{i,n} f(t_i),$$

wobei die Konstanten

$$\alpha_{i,n} := \int_a^b L_{i,n}(t)\, dt$$

nur von der Wahl der Knoten $t_0, \ldots, t_n$ abhängen.

Wir sehen wieder, daß die Approximation $\hat{I}$ linear in f ist, aber im allgemeinen nicht additiv. Zur Erfüllung der Eigenschaft (III) muß man notwendig fordern, daß

$$\alpha_{i,n} > 0 \quad \text{für alle} \quad i, n$$

Für den Spezialfall *äquidistanter Knoten*

$$h_i = h = \frac{b-a}{n}, \quad t_i = ih, \quad i = 0, \ldots, n$$

vereinfacht sich der Ausdruck für die $\alpha_{i,n}$ durch Substitution $s := (t-a)/h$ zu

$$\begin{aligned}
\alpha_{i,n} &= \int_a^b \frac{(t-t_0)\cdots(t-t_{i-1})(t-t_{i+1})\cdots(t-t_n)}{(t_i-t_0)\cdots(t_i-t_{i-1})(t_i-t_{i+1})\cdots(t_i-t_n)}\, dt \\
&= h \underbrace{\int_0^n \frac{(s-0)\cdots(s-(i-1))(s-(i+1))\cdots(s-n)}{(i-0)\cdots(i-(i-1))(i-(i+1))\cdots(i-n)}\, ds}_{=:\, n\cdot\bar{\alpha}_{i,n} \text{ unabhängig von } h,a,b} \\
&= \frac{b-a}{n} \cdot n \cdot \bar{\alpha}_{i,n} \\
&= (b-a)\bar{\alpha}_{i,n}
\end{aligned}$$

Wir gelangen so zu den klassischen *Newton-Cotes-Formeln*

$$\hat{I}(f) = (b-a) \sum_{i=0}^{n} \bar{\alpha}_{i,n} f(a+ih)$$

Die Werte $\bar{\alpha}_{i,n}$ müssen nur einmal berechnet bzw. eingegeben werden. Ferner läßt sich leicht zeigen (s. Aufgabe ?), daß

$$\text{a)} \quad \sum_{i=0}^{n} \bar{\alpha}_{i,n} = 1$$

$$\text{b)} \quad \bar{\alpha}_{n-i,n} = \bar{\alpha}_{i,n} \text{ für } i = 0, \ldots, n$$

Bemerkung 6.1.1 *Falls einige Koeffizienten $\bar{\alpha}_{i,n}$ negativ sind, so ist die Abweichung*

$$\sum_{i=0}^{n} |\bar{\alpha}_{i,n}| - 1$$

ein natürliches Maß für die Verletzung der Bedingung a).

Tabelle: $\bar{\alpha}_{i,n}$ $(i = 0, \ldots, n)$

$n\backslash i$	0	1	2	3	4	5	6	
1	$\frac{1}{2}$	$\frac{1}{2}$						Trapezregel
2	$\frac{1}{6}$	$\frac{4}{6}$	$\frac{1}{6}$					Simpson-Regel
3	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$				Faßregel
4	$\frac{7}{90}$	$\frac{32}{90}$	$\frac{12}{90}$	$\frac{32}{90}$	$\frac{7}{90}$			Milne-Regel
5								
6	$\frac{41}{840}$	$\frac{216}{840}$	$\frac{27}{840}$	$\frac{272}{840}$	$\frac{27}{840}$	$\frac{216}{840}$	$\frac{41}{840}$	Weddle-Regel

6.2 Klassische Romberg-Quadratur

6.2.1 Asymptotische Entwicklung der Trapezsumme

Im folgenden wird die Struktur des Approximationsfehlers der Trapezregel als Funktion in h näher analysiert. Er besagt, daß sich die Trapezsumme $T(h)$ in einer sogenannten *asymptotischen Entwicklung* in h^2 entwickeln läßt.

Satz 6.2.1 *Sei $f \in C^{2m+1}[a,b]$ und $h = \frac{b-a}{n}$ für ein $n \in \mathbf{N} - \{0\}$. Dann besitzt die Trapezsumme $T(h)$ folgende asymptotische Entwicklung des Approximationsfehlers*

$$T(h) = \int_a^b f(t)\,dt + \tau_2 h^2 + \tau_4 h^4 + \cdots + \tau_{2m} h^{2m} + R_{2m+2}(h) h^{2m+2} \tag{6.2.1}$$

wobei

a) $\tau_{2k} = \dfrac{B_{2k}}{(2k)!}(f^{(2k-1)}(b) - f^{(2k-1)}(a))$ *mit den* Bernoulli-Zahlen B_{2k}

b) $R_{2m+2}(h) = -\displaystyle\int_a^b K_{2m+2}(t;h) f^{(2m)}(t)\,dt$ *mit dem* Bernoulli-Polynom *zweiter Art (s. u.)* $K_{2m+2}(t;h)$

Der Restterm $R_{2m+2}(h)$ *ist gleichmäßig beschränkt in* h, *d.h. es gibt ein* $C_{2m+2} \geq 0$, *so daß*

$$|R_{2m+2}(h)| \leq C_{2m+2}|b-a| \quad \textit{für alle} \quad 0 \leq h \leq b-a$$

Bemerkung 6.2.2

1) Für große k gilt

$$B_{2k} \approx (2k)!$$

Auch für analytische Funktionen $f \in C^{\omega}[a,b]$ *divergiert* die Reihe (6.2.1) im allgemeinen für $m \to \infty$.

2) Für periodische Funktionen f mit Periode $b-a$ verschwinden alle τ_{2k}, d.h. der gesamte Fehler bleibt im Restterm. In diesem Fall kann mit der Extrapolation keine Verbesserung erzielt werden. Tatsächlich ist dann sogar die Trapezsumme selbst optimal.

Im Gegensatz zu den aus der Analysis bekannten Reihenentwicklungen (Taylor-, Fourier-Reihen) wird in Satz 6.2.1 die Funktion in eine *divergenten Reihe* entwickelt. Dies scheint zunächst wenig Sinn zu machen; tatsächlich können jedoch häufig die endlichen Partialsummen genutzt werden, um einen Funktionswert hinlänglich genau zu berechnen, obwohl die zugehörige unendliche Reihe divergiert. Zur Illustration, daß eine derartige Entwicklung in eine divergente Reihe dennoch numerisch nützlich sein kann, betrachten wir folgendes Beispiel (siehe [14]).

Beispiel 6.2.3 Sei $f(h)$ eine Funktion mit einer asymptotischen Entwicklung in h, so daß für alle $h \in \mathbf{R}$ und $n \in \mathbf{N}$

$$f(h) = \sum_{k=0}^{n}(-1)^k k! \cdot h^k + \theta(-1)^{n+1}(n+1)!\, h^{n+1} \quad \text{für ein} \quad 0 < \theta = \theta(x) < 1 \qquad (6.2.2)$$

Die Reihe $\sum(-1)^k k!\, h^k$ divergiert für alle $h \neq 0$. Betrachte man die Partialsummenfolge

$$s_n(h) := \sum_{k=0}^{n}(-1)^k k!\, h^k$$

für kleine h, $0 \neq h \ll 1$, so scheint diese zunächst zu konvergieren, da die Glieder $(-1)^k k!\, h^k$ der Reihe zunächst stark abnehmen. Ab einem gewissen Index überwiegt jedoch $k!$, die Glieder werden beliebig groß und die Partialsummenfolge divergiert. Da

$$|f(h) - s_n(h)| = |(-1)^{n+1}\theta(n+1)!\, h^{n+1}| < |(n+1)!\, h^{n+1}|$$

ist der Fehler, den wir bei der Approximation von f durch s_n machen, stets kleiner als das erste weggelassene Glied. Um $f(h)$ auf eine (absolute) Genauigkeit von eps zu bestimmen, müssen wir ein n finden, so daß

$$|(n+1)!\, h^{n+1}| < \text{ eps} \qquad (6.2.3)$$

Ganz konkret erhalten wir z.B. $f(\frac{1}{1000})$ auf zehn Dezimalstellen genau für $n = 3$ durch

$$s_3(10^{-3}) = 1 - 10^{-3} + 2 \cdot 10^{-6} - 6 \cdot 10^{-9}$$

Wegen ihres "fast konvergenten" Verhaltens nannte Legendre solche Reihen auch *semikonvergent.* Euler machte sich das Leben etwas einfacher, indem er für alle Reihen dieselbe Schreibweise benutzte, ob sie nun konvergierten oder nicht.

Der folgende Beweis von Satz 6.2.1 benutzt das klassische Resultat der Eulerschen Summenformel. Diese Beweistechnik ist nicht verallgemeinerbar auf den Fall von Differentialgleichungen. Es sei deshalb dem Leser anheim gestellt, den Beweis zu überspringen.

Wir beweisen zunächst die Euler'schen Summenformel und leiten daraus die asymptotische Entwicklung der Trapezsumme ab. In der Darstellung richten wir uns stark nach der in [14], XIV. Kapitel. Die Euler'sche Summenformel stellt einen Zusammenhang her zwischen der Summe

$$f_0 + \cdots + f_n, \quad f_i := f(i)$$

der Funktionswerte einer Funktion f an den Stellen $i = 0, \ldots, n$ und dem Integral

$$\int_0^n f(x)\, dx$$

Satz 6.2.4 *(Euler'sche Summenformel) Sei* $f \in C^{2k+1}[0,n]$. *Dann gilt*

$$\sum_{i=0}^{n} f_i = \int_0^n f(x)\, dx + \frac{1}{2}(f_n + f_0) + \sum_{i=1}^{k} \frac{B_{2i}}{(2i)!}(f_n^{(2i-1)} - f_0^{(2i-1)}) + R_k$$

mit dem Restterm

$$R_k = \int_0^n P_{2k+1}(x) f^{(2k+1)}(x)\, dx$$

Dabei sind

a) B_i *die* Bernoulli-Zahlen, *definiert durch*

$$B_0 := 0 \quad \textit{und} \quad (B+1)^n - B^n := \sum_{i=0}^{n} \binom{n}{i} B_i - B_n = 0$$

b) P_k *die* Bernoulli-Polynome, *1-periodisch mit*

$$P_k(x) := \frac{1}{k!}(x+B)^k := \frac{1}{k!} \sum_{i=0}^{k} \binom{k}{i} B_i x^{k-i} \quad \textit{für} \quad 0 \le x < 1$$

c) $f_i^{(k)} := f^{(k)}(i)$ *für* $i, k \in \mathbf{N}$

Wir beweisen zunächst folgendes Lemma:

Lemma 6.2.5 *Für* $f \in C^1[0,n]$ *gilt*

$$\sum_{i=0}^{n} f_i = \int_0^n f(x)\, dx + \frac{1}{2}(f_0 + f_n) + \int_0^n (x - [x] - \frac{1}{2}) f'(x)\, dx$$

Beweis: Es ist

$$\int_{i-1}^{i} f'(x)\,dx = f_i - f_{i-1} \quad \text{für } i = 1,2,\ldots$$

und daher

$$\sum_{i=1}^{n} \int_{i-1}^{i} if'(x)\,dx = \sum_{i=1}^{n} f(f_i - f_{i-1}) = -\sum_{i=0}^{n} f_i + (n+1)f_n$$

Für alle $x \in [i-1,i]$ gilt $i = [x]+1$, also

$$\sum_{i=0}^{n} f_i = (n+1)f_n - \int_0^n ([x]+1)f'(x)\,dx \tag{6.2.4}$$

Mit partieller Integration folgt

$$\int_0^n xf'(x)\,dx = [xf(x)]_0^n - \int_0^n f(x)\,dx = nf(n) - \int_0^n f(x)\,dx$$

Eingesetzt in (6.4) ergibt sich

$$\sum_{i=0}^{n} f_i = \int_0^n f(x)\,dx + f_n + \int_0^n (x - [x] - \frac{1}{2})f'(x)\,dx$$

und mit $\int_0^n \frac{1}{2}f'(x) = \frac{1}{2}(f_n - f_0)$ die Behauptung. □

Die Funktion

$$P_1(x) := x - [x] - \frac{1}{2}$$

ist das *erste Bernoulli-Polynom.* Sie ist 1-periodisch, fast überall stetig und besitzt die Fourier-Entwicklung

$$P_1(x) = -\sum_{n=1}^{\infty} \frac{\sin\ 2\pi nx}{n\pi}$$

Damit können wir leicht die übrigen Bernoulli-Polynome definieren.

Definition 6.2.6 *Für $i = 1,2,\ldots$ sind die* Bernoulli-Polynome P_i *definiert durch*

$$\begin{aligned} P_{2k}(x) &:= (-1)^{k-1} \sum_{n=1}^{\infty} \frac{2\ \cos\ 2\pi nx}{(2\pi n)^{2k}} \\ P_{2k+1}(x) &:= (-1)^{k-1} \sum_{n=1}^{\infty} \frac{2\ \sin\ 2\pi nx}{(2\pi n)^{2k+1}} \end{aligned}$$

Daraus lassen sich folgende Eigenschaften ableiten (Beweis zur Übung).

Bemerkung 6.2.7

1) *Die P_i sind fast überall C^1, 1-periodisch, und es gilt $P'_{i+1}(x) = P_i(x)$.*

2) $P_{2k+1}(0) = 0$

3) $P_{2k}(0) = (-1)^{k-1} \sum_{n=1}^{\infty} \frac{2}{(2\pi n)^{2k}} = \frac{B_{2k}}{(2k)!}$

4) $P_i(x) = \frac{1}{k!}(x+B)^k := \frac{1}{k!} \sum_{i=0}^{k} \binom{k}{i} B_i x^{k-i}$ *für* $0 \le x < 1$

Die ersten Werte für die Bernoulli-Zahlen B_i sind:

i	0	1	2	3	4	5	6	7
B_i	1	$-\frac{1}{2}$	$\frac{1}{6}$	0	$-\frac{1}{30}$	0	$\frac{1}{42}$	0

i	8	9	10	11	12	13	14
B_i	$-\frac{1}{30}$	0	$\frac{5}{66}$	0	$-\frac{691}{2730}$	0	$\frac{7}{6}$

Beweis von Satz 6.2.4 Nach Lemma 6.2.5 gilt

$$\sum_{i=0}^{n} f_i = \int_0^n f(x)\,dx + \frac{1}{2}(f_0 + f_n) + \int_0^n P_1(x)f'(x)\,dx \tag{6.2.5}$$

Mit partieller Integration erhalten wir

$$\begin{aligned}\int_0^n P_1(x)f'(x)\,dx &= [P_2 f']_0^n - \int_0^n P_2 f''dx \\ &= \frac{B_2}{2!}(f'_n - f'_0) - [P_3 f'']_0^n + \int_0^n P_3 f'''dx \\ &= \frac{B_2}{2!}(f'_n - f'_0) + \int_0^n P_3 f'''dx\end{aligned}$$

oder allgemeiner

$$\int_0^n P_{2k-1} f^{(2k-1)}dx = \frac{B_{2k}}{(2k)!}(f_n^{(2k-1)} - f_0^{(2k-1)}) + \int_0^n P_{2k+1} f^{(2k+1)}dx$$

Eingesetzt in 6.2.5 ergibt sich damit die Behauptung. □

Bemerkung 6.2.8 *Unter den Voraussetzungen des Satzes gilt*

$$R_k = -\int_0^n \underbrace{(P_{2k+2}(x) - \frac{B_{2k+2}}{(2k+2)!})}_{=:K_{2k+2}(x)} f^{(2k+2)}(x)\,dx$$

Beweis: Partielle Integration liefert

$$\begin{aligned}R_k &= \int_0^n P_{2k+1}(x)f^{(2k+1)}(x)\,dx \\ &= [P_{2k+2}(x)f^{(2k+1)}(x)]_0^n - \int_0^n P_{2k+2}(x)f^{(2k+2)}(x)\,dx \\ &= -\int_0^n (P_{2k+2}(x) - \frac{B_{2k+2}}{(2k+2)!})f^{(2k+2)}(x)\,dx\end{aligned}$$

da $P_{2k+2}(0) = P_{2k+2}(n) = \frac{B_{2k+2}}{(2k+2)!}$. □

Aus der Euler'schen Summenformel können wir jetzt die asymptotische Entwicklung der Trapezregel ableiten.

Beweis von Satz 6.2.4 Mit $h = (b-a)/n$ gilt für die Trapezregel

$$T(h) = h[\frac{1}{2}(f(a)+f(b)) + \sum_{i=1}^{n-1} f(a+ih)]$$

Um die Aussage auf die Euler'sche Summenformel zurückzuführen, transformieren wir das Intervall $[0,n]$ auf $[a,b]$

$$\begin{array}{rcl} [0,n] & \to & [a,b] \\ x & \mapsto & t = a+hx \end{array}$$

und definieren $g \in C^{2k+1}[0,n]$ durch

$$g : [0,n] \to \mathbf{R}\,, \quad g(x) := f(a+hx)$$

Damit gilt

a) $g^{(j)}(x) = f^{(j)}(a+hx)h^j = f^{(j)}(t)h^j$

b) $\int_a^b f(t)\,dt = h\int_0^n g(x)\,dx$

c) $T(h) = h\left(\frac{1}{2}(g_0+g_n) + \sum_{i=1}^{n-1} g_i\right), \quad g_i = g(i) = f(a+ih)$

Setzen wir g in die Euler'sche Summenformel ein und multiplizieren beide Seiten mit h, so folgt

$$T(h) - \int_a^b f(t)\,dt = \sum_{i=1}^{k} \frac{B_{2i}}{(2i)!}(f^{(2i-1)}(b) - f^{(2i-1)}(a))h^{2i} + \tilde{R}_k$$

wobei nach Bemerkung 6.2.8

$$\begin{aligned} \tilde{R}_k &= h \cdot R_k \\ &= -h\int_0^n K_{2k+2}(x) g^{(2k+2)}(x)\,dx \\ &= -h^{2k+3}\int_0^n K_{2k+2}(x) f^{(2k+2)}(a+hx)\,dx \\ &= -h^{2k+2}\int_a^b \underbrace{K_{2k+2}(\frac{t-a}{h})}_{=:K_{2k+2}(t;h)} f^{(2k+2)}(t)\,dt\,. \end{aligned}$$

□

6.2.2 Grundidee der Extrapolation

Wir haben in 6.1 das Integral

$$I(f) := \int_a^b f(t)\,dt$$

approximiert durch die Trapezsumme

$$T(h) := T_n := h\left(\frac{1}{2}(f(a)+f(b)) + \sum_{i=1}^{n-1} f(a+ih)\right) \quad \text{mit} \quad h = \frac{b-a}{n},$$

wobei die Güte der Approximation von der Schrittweite h abhängt. Für $h \to 0$ konvergiert der Ausdruck $T(h)$ gegen das Integral. Genauer müßten wir sagen "für $n \to \infty$ und $h = (b-a)/n$", da $T(h)$ nur für die diskreten Werte $h = (b-a)/n$, $n = 1,2,\ldots,$ erklärt ist.

$$\lim_{h\to 0} T(h) := \lim_{n\to\infty} T_n = I(f) \tag{6.2.6}$$

Zur Darstellung der Grundidee der Romberg-Quadratur gehen wir zunächst davon aus, daß wir $T(h)$ für zwei Werte

$$h_i := \frac{b-a}{n_i}, \quad i = 1,2$$

berechnet haben.

Beispiel 6.2.9 $f(t) = t^2$, $a = 0$, $b = 1$, $n_1 = 1$, $n_2 = 2$

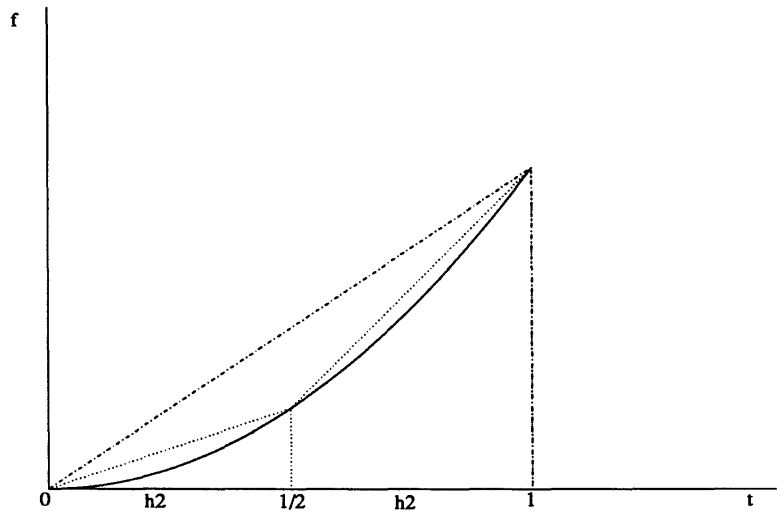


Abbildung 6.22: Trapezsummen $T(h_1)$ und $T(h_2)$ für $f(t) = t^2$.

$$\begin{aligned} T(h_1) &= T(1) = \frac{1}{2}(f(0)+f(1)) = \frac{1}{2} \\ T(h_2) &= T(\tfrac{1}{2}) = \frac{1}{2}(\frac{1}{2}(f(0)+f(1)) + f(\tfrac{1}{2})) = \frac{1}{2}(\frac{1}{2}+\frac{1}{4}) = \frac{3}{8} \end{aligned}$$

Wegen $R_4(h) = 0$ gilt im vorliegenden Fall:

$$\begin{aligned} T(h_1) &= I(f) + \tau_2 h_1^2 \\ T(h_2) &= I(f) + \tau_2 h_2^2 \end{aligned}$$

Damit können wir den Koeffizienten τ_2 bestimmen durch

$$\tau_2 = \frac{T(h_1) - T(h_2)}{h_1^2 - h_2^2}$$

und erhalten als Näherung für $I(f)$

$$I(f) = T(h_1) - \tau_2 h_1^2 = T(h_1) - \frac{T(h_2) - T(h_1)}{h_2^2 - h_1^2} h_1^2 =: T_{22} \tag{6.2.7}$$

In obigem Beispiel folgt

$$\begin{aligned} \tau_2 &= \frac{\frac{3}{8} - \frac{1}{2}}{\frac{1}{4} - 1} = \frac{1}{8} \cdot \frac{4}{3} = \frac{1}{6} \\ I(f) &= T(h_1) - \tau_2 h_1^2 = \frac{1}{2} - \frac{1}{6} = \frac{1}{3} = \int_0^1 t^2 \, dt \end{aligned}$$

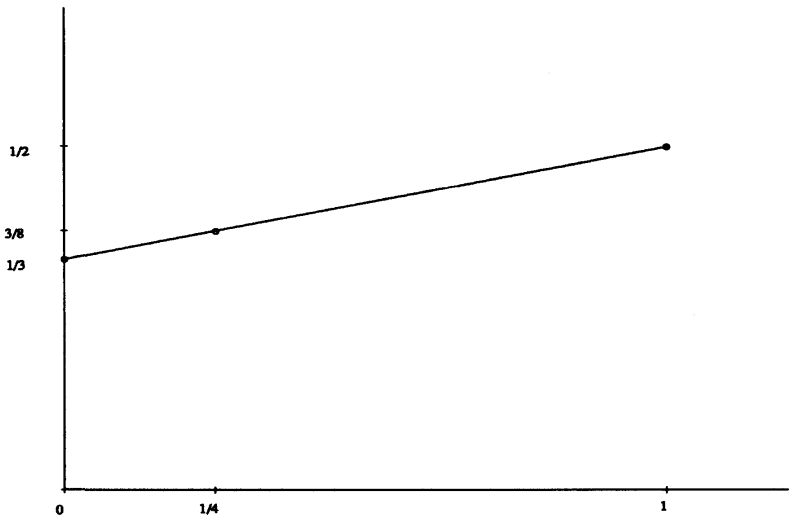


Abbildung 6.23: (lineare) Extrapolation.

Das oben gewählte Verfahren können wir für $f \neq t^2$ näherungsweise und wiederholt durchführen; dabei werden sukzessive die Entwicklungskoeffizienten $\tau_2, \tau_4, \ldots$ eliminiert. Wir können die Formel 6.2.7 auch wie folgt erklären: Aus der Basis der asymptotischen Entwicklung der Trapezregel bestimmen wir das Interpolationspolynom in h^2 zu den Punkten $(h_1^2, T(h_1))$ und $(h_2^2, T(h_2)$

$$P(h^2; h_1^2, h_2^2) = T(h_1) + \frac{T(h_2) - T(h_1)}{h_2^2 - h_1^2}(h^2 - h_1^2)$$

und extrapolieren für $h^2 = 0$, d. h.

$$T_{22} = P(0; h_1^2, h_2^2) = T(h_1) - \frac{T(h_2) - T(h_1)}{h_2^2 - h_1^2} h_1^2$$

Als extrapolierten Wert für $h^2 = 0$ erwarten wir eine bessere Näherung von $I(f)$ erwarten.

6.2.3 Extrapolationsverfahren

Die im letzten Abschnitt beschriebene Grundidee führt zur Klasse der sogenannten *Extrapolationsmethoden.* Sie können immer dann benutzt werden, wenn ein Verfahren eine asymptotische Entwicklung des Approximationsfehlers zuläßt. Um das zu präzisieren, gehen wir aus von einem Verfahren $T(h)$, welches in Abhängigkeit von einer "Schrittweite" h einen gesuchten Wert τ_0 berechnet. (Dabei lassen wir zu, daß $T(h)$ nur für diskrete Werte h definiert ist (s.o.)). Zusätzlich verlangen wir, daß das Verfahren für $h \to 0$ gegen τ_0 konvergiert, d.h.

$$\lim_{h \to 0} T(h) = \tau_0 \tag{6.2.8}$$

Definition 6.2.10 *Das Verfahren $T(h)$ zur Berechnung von τ_0 besitzt eine* asymptotische Entwicklung in h^p, $p \in \{0,1\}$, bis zur Ordnung pm, *falls es Konstanten $\tau_p, \tau_{2p}, \ldots, \tau_{mp} \in \mathbf{R}$ gibt, so daß*

$$t(h) = \tau_0 + \tau_p h^p + \tau_{2p} h^{2p} + \cdots + \tau_{mp} h^{mp} + O(h^{(m+1)p}) \quad \textit{für} \quad h \to 0 \tag{6.2.9}$$

Bemerkung 6.2.11 *Die Trapezregel besitzt nach Satz 6.2.1 für Funktionen $f \in C^{2m+1}[a,b]$ eine asymptotische Entwicklung in h^2 bis zur Ordnung $2m$.*

Haben wir $T(h)$ für k verschiedene Schrittweiten

$$h = h_{i-k+1}, \ldots, h_i$$

berechnet, so können wir das Interpolationspolynom in h^p

$$P_{ik}(h^p) = P(h^p; h^p_{i-k+1}, \ldots, h^p_i) \in \mathbf{P}_{k-1}(h^p)$$

zu den Stützpunkten

$$(h^p_{i-k+1}, T(h_{i-k+1})), \ldots, (h^p_i, T(h_i))$$

bestimmen und durch Auswertung an der Stelle $h = 0$ extrapolieren. Wir erhalten so die Näherung T_{ik} von τ_0

$$T_{ik} := P_{ik}(0) \quad \text{für} \quad 1 \le k \le i$$

Zur Berechnung von T_{ik} verwenden wir natürlich den Algorithmus von Aitken und Neville. Die Rekursionsformel (5...) transformiert sich dabei wie folgt in die jetzige Situation.

$$\begin{aligned} &\text{a)} \quad T_{i1} := T(h_i) \quad \text{für} \quad i = 1, 2, \ldots \\ &\text{b)} \quad T_{ik} := T_{i,k-1} + \frac{P_{i,k-1} - P_{i-1,k-1}}{\left(\frac{h_{i-k+1}}{h_i}\right)^p - 1} \quad \text{für} \quad 2 \le k \le i \end{aligned} \tag{6.2.10}$$

Bemerkung 6.2.12 In Anlehnung an [7] zählen wir bei den Extrapolationsmethoden von 1 an. Dies führt, wie wir unten sehen werden, zu einem angenehmeren Zusammenhang zwischen der Anzahl k der berechneten Werte $T(h_1), \ldots, T(h_k)$ und der Ordnung der Approximation T_{kk}. T_{kk} ist der extrapolierte Wert für die k Punkte $T_{11}, \ldots, T_{k1}$.

Das Schema von Neville geht dabei über in das sogenannte *Extrapolationstableau*

$$
\begin{array}{lllllll}
T_{11} & & & & & & \\
 & \searrow & & & & & \\
T_{21} & \rightarrow & T_{22} & & & & \\
 & \searrow & & \searrow & & & \\
T_{31} & \rightarrow & T_{32} & \rightarrow & T_{33} & & \\
\vdots & & & & & \ddots & \\
\vdots & & & & & & \ddots \\
T_{k-1,1} & \rightarrow & \cdots & & & \rightarrow & T_{k-1,k-1} \\
 & \searrow & & & & & \\
T_{k1} & \rightarrow & \cdots & & & \rightarrow & T_{k,k-1}
\end{array}
\qquad
\begin{array}{l}
 \\
 \\
 \\
 \\
 \\
 \\
 \\
\searrow \\
\rightarrow T_{kk}
\end{array}
$$

Bezeichnen wir die Approximationsfehler der durch Extrapolation gewonnenen Näherungen T_{ik} von τ_0 mit

$$\varepsilon_{ik} := |T_{ik} - \tau_0| \quad \text{für} \quad 1 \leq k \leq i,$$

so können wir diese entsprechend in dem *Fehlertableau* anordnen:

$$
\begin{array}{lllll}
\varepsilon_{11} & & & & \\
\varepsilon_{21} & \varepsilon_{22} & & & \\
\varepsilon_{31} & \varepsilon_{32} & \varepsilon_{33} & & \\
\vdots & & & \ddots & \\
\varepsilon_{k-1,1} & \cdots & \cdots & \cdots & \varepsilon_{k-1,k-1} \\
\varepsilon_{k1} & \cdots & \cdots & \cdots & \varepsilon_{k,k-1} \qquad \varepsilon_{kk}
\end{array}
$$

Der folgende Satz gibt Auskunft darüber, wie sich diese Fehler im wesentlichen verhalten.

Satz 6.2.13 *(BULIRSCH, 1964) Sei $T(h)$ ein Verfahren mit einer asymptotischen Entwicklung (6.2...) in h^p bis zur Ordnung m und $h_1, \ldots, h_m$ m verschiedene Schrittweiten. Dann gilt für die Approximationsfehler ε_{ik} der Extrapolationswerte T_{ik} in führender Ordnung*

$$\varepsilon_{ik} \doteq |\tau_{kp}| \underbrace{h_{i-k+1}^p \cdots h_i^p}_{k \text{ Faktoren}} \quad \text{für} \quad 1 \leq k \leq i \leq m$$

Genauer gilt

$$\varepsilon_{ik} = |\tau_{kp}| \, h_{i-k+1}^p \cdots h_i^p + \sum_{j=i-k+1}^{i} O(h_j^{(k+1)p}) \quad \text{für} \quad h_j \leq h \to 0$$

Der Satz besagt, daß wir im wesentlichen pro Spalte des Extrapolationstableaus p Ordnungen gewinnen können. Da wir es jedoch mit asymptotischen und nicht mit Reihenentwicklungen zu tun haben, ist diese Sichtweise zu optimistisch. Warnend sei jedoch vorab erwähnt, daß die hohe Ordnung wenig nützt, wenn die Restterme der asymptotischen Entwicklung, die sich hinter dem $O(h_j^{(k+1)_p})$ verbergen, sehr groß werden. Zum Beweis des Satzes verwenden wir folgende Hilfsaussage

Lemma 6.2.14 *Für die Lagrange-Funktionen $L_0, \ldots, L_n$ zu den Stützstellen $t_0, \ldots, t_n$ gilt*

$$\sum_{j=0}^{n} L_j(0) t_j^m = \begin{cases} 1 & \text{für } m = 0 \\ 0 & \text{für } 1 \le m \le n \\ (-1)^n t_0 \cdot \cdots \cdot t_n & \text{für } m = n+1 \end{cases}$$

Beweis: Für $0 \le m \le n$ ist $P(t) = t^m$ das Interpolationspolynom zu den Punkten (t_j, t_j^m) für $j = 0, \ldots, n$ und daher

$$P(t) = t^m = \sum_{j=0}^{n} L_j(t) P(t_j) = \sum_{j=0}^{n} L_j(t) t_j^m$$

Setzen wir $t = 0$, so folgt die Behauptung für die ersten beiden Fälle. Für den Fall $m = n+1$ betrachten wir das Polynom

$$Q(t) := t^{n+1} - \sum_{j=0}^{n} L_j(t) t_j^{n+1}$$

Dies ist ein normiertes Polynom vom Grad $n+1$ mit den Nullstellen $t_0, \ldots, t_n$, also

$$Q(t) = (t - t_0) \cdot \cdots \cdot (t - t_n)$$

und speziell

$$\sum_{j=0}^{n} L_j(0) t_j^{n+1} = -Q(0) = (-1)^n t_0 \cdot \cdots \cdot t_n \, .$$

□

Beweis von Satz 6.2.13 Da $T(h)$ eine asymptotische Entwicklung in h^p bis zur Ordnung m besitzt, gilt für $1 \le k \le m$

$$T_{j1} = T(h_j) = \tau_0 + \tau_p h_j^p + \cdots + \tau_{(k+1)_p} h_j^{kp} + O(h_j^{(k+1)_p}) \qquad (6.2.11)$$

Es reicht aus, die Behauptung für $i = k$ zu zeigen (der Fall $i \neq k$ folgt daraus durch Verschieben der Indizes der h_j). Seien also $P_{kk} = P(h^p; h_1^p, \ldots, h_k^p)$ das Interpolationspolynom in h^p zu den Stützpunkten $(h_j^p, T(h_j))$ für $j = 1, \ldots, k$ und $L_1(h^p), \ldots, L_k(h^p)$ die Lagrange-Polynome zu den Stützstellen $h_1^p, \ldots, h_k^p$. Dann gilt

$$P_{kk}(h^p) = \sum_{j=1}^{k} L_j(h^p) T_{j1}$$

Daher ist wegen 6.2.11 und Lemma 6.2.14

$$\begin{aligned} T_{kk} &= P_{kk}(0) = \sum_{j=1}^{k} L_j(0) T_{j1} \\ &= \sum_{j=1}^{k} L_j(0) \left[\tau_0 + \tau_p h_j^p + \cdots + \tau_{kp} h_j^{kp} + O(h_j^{(k+1)p})\right] \\ &= \tau_0 + \tau_{kp}(-1)^{k-1} h_1^p \cdot \cdots \cdot h_k^p + \sum_{j=1}^{k} O(h_j^{(k+1)p}) \end{aligned}$$

also

$$\varepsilon_{kk} = |T_{kk} - \tau_0| = |\tau_{kp}|\ h_1^p \cdot \cdots \cdot h_k^0 + \sum_{j=1}^{k} O(h_j^{(k+1)p})\ .$$

□

Die dargestellte Theorie legt für ein Verfahren $T(h)$ mit asymptotischer Entwicklung den folgenden *Extrapolationsalgorithmus* nahe. Dabei gehen wir aus von der *Grundschrittweite* (engl.: *basic stepsize*) H und bilden die Schrittweiten h_i durch Teilen von H.

H	:=	$b - a > 0$	Grundschrittweite (basic stepsize)
h_i	:=	H/n_i	innere Schrittweiten (internal stepsizes)

Algorithmus 6.2.15 *Extrapolationsverfahren*

1. *Wähle für eine gegebene Grundschrittweite H eine Folge von von Schrittweiten $h_1, h_2 \ldots$ mit $h_j = H/n_j$, $n_{j+1} > n_j$ und setze $i := 1$*
2. *Bestimme $T_{i1} = T(h_i)$*
3. *Berechne T_{ik} für $k = 2, \ldots, i$ mit dem Schema von Neville*

$$T_{ik} = T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{\left(\frac{n_i}{n_{i-k+1}}\right)^p - 1}$$

4. *Falls T_{ii} genau genug oder i zu groß, beende den Algorithmus. Ansonsten erhöhe i um 1 und gehe zurück zu 2.*

Diese grobe Beschreibung läßt natürlich noch viele Fragen offen. So ist nicht klar, was "T_{ii} genau genug" oder "i zu groß" heißen soll. Auch ist nicht klar, wie die Schrittweiten h_j gewählt werden sollen. Am Beispiel der Romberg-Quadratur werden wir in den nächsten Abschnitten darauf näher eingehen.

6.2.4 Details des Algorithmus

Wie wir oben gesehen haben, ist die Trapezsumme ein Verfahren mit einer asymptotischen Entwicklung in h^2, wobei die Ordnung, bis zu der wir entwickeln können, von der Glattheit des Integranden abhängt. Daher können wir alles im letzten Abschnitt

beschriebene auf die Trapezregel anwenden und erhalten als Extrapolationsverfahren die von Romberg eingeführte klassische *Romberg-Quadratur.*

Der *Aufwand* A_i zur Berechnung von T_{ii} läßt sich im wesentlichen durch die Anzahl der benötigten Funktionsauswertungen von f messen.

$$A_i := \text{Anzahl der zur Berechnung von } T_{ii} \text{ benötigten } f\text{-Auswertungen.}$$

Diese Zahlen können je nachdem wie wir die Folge $n_1, n_2, \ldots$ wählen, verschieden sein. Wir ordnen daher jeder aufsteigenden Folge

$$\mathcal{F} = \{n_1, n_2, \ldots\}, \quad n_i \in \mathbf{N} - \{0\}$$

die zugehörige Aufwandsfolge

$$\mathcal{A} = \{A_1, A_2, \ldots\}$$

zu. Für die sogenannte *Romberg-Folge* ergibt sich

$$\begin{aligned} \mathcal{F}_R &:= \{1,2,4,8,16,\ldots\}, \quad n_i = 2^{i-1} \\ \mathcal{A}_R &:= \{2,3,5,9,17,\ldots\}, \quad A_i = n_i + 1 \end{aligned}$$

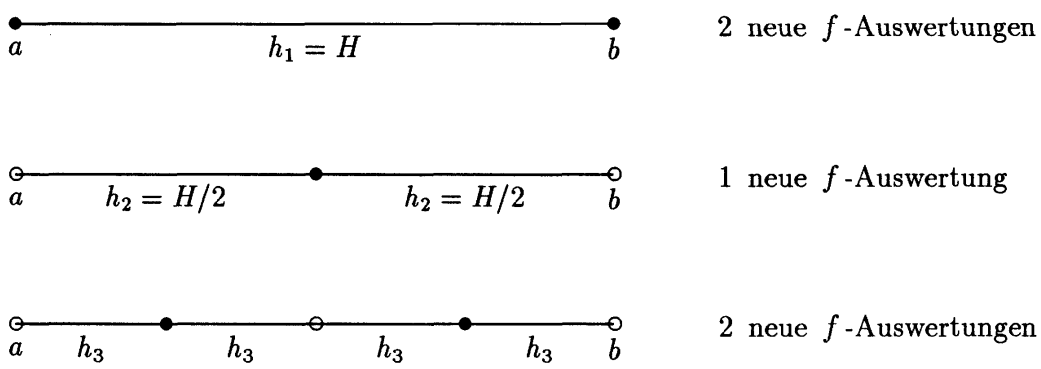


Abbildung 6.24: Berechnung der Trapezsummen bei der Romberg–Folge.

Für diese Folge lassen sich die Trapezsummen besonders einfach rekursiv berechnen.

$$\begin{aligned} T_{11} &= \frac{H}{2}(f(a) + f(a+H)) \\ T_{21} &= \frac{H}{2}(\frac{1}{2}f(a) + f(a+\frac{H}{2}) + \frac{1}{2}f(a+H)) = \frac{1}{2}(T_{11} + Hf(a+\frac{H}{2})) \\ T_{31} &= \frac{1}{2}(T_{21} + \frac{H}{2}(f(a+\frac{H}{4}) + f(a+\frac{3H}{4}))) \\ &\vdots \qquad \vdots \end{aligned}$$

Stelle man die Extrapolationswerte T_{ii} der Romberg-Quadratur als Quadraturformel dar

$$T_{ii} = H \sum_{j=1}^{A_i} \bar{\alpha}_j f_j,$$

wobei f_j der j-te berechnete Funktionswert ist, so läßt sich zeigen (siehe Aufgabe 6...), daß sich für die Romberg-Folge $\mathcal{F}_R$ nur positive Gewichte α_j ergeben. Vom Aufwand her gibt es noch eine günstigere Folge, die sogenannte *Bulirsch-Folge* (die jedoch schon bei Romberg vorkommt):

$$\mathcal{F}_B := \{1,2,3,4,6,8,12,16,24,\ldots\}, \quad n_i = \begin{cases} 2^{k-1} & \text{falls } i = 2k \\ 3 \cdot 2^{k-1} & \text{falls } i = 2k+1 \\ 1 & \text{falls } i = 1 \end{cases}$$

Für diese Folge gilt

$$\mathcal{A}_B = \{2,3,5,7,9,13,17,\ldots\}.$$

Allerdings hat sie den Nachteil, daß die zugehörige Quadraturformel auch negative Gewichte enthalten kann.

	0	$\frac{1}{8}$	$\frac{1}{6}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{3}{8}$	$\frac{1}{2}$
$T_{1,1}$	$\frac{1}{2}$						
$T_{2,1}$	$\frac{1}{4}$						$\frac{1}{2}$
$T_{2,2}$	$\frac{1}{6}$						$\frac{2}{3}$
$T_{3,2}$	$\frac{1}{10}$				$\frac{3}{5}$		$\frac{2}{3}$
$T_{3,3}$	$\frac{11}{120}$				$\frac{27}{40}$		$-\frac{8}{15}$
$T_{4,3}$	$\frac{13}{210}$			$\frac{16}{21}$	$-\frac{27}{35}$		$\frac{94}{105}$
$T_{4,4}$	$\frac{151}{2520}$			$\frac{256}{315}$	$-\frac{243}{280}$		$\frac{104}{105}$
$T_{5,4}$	$\frac{17}{420}$		$\frac{9}{20}$	$-\frac{64}{105}$	$\frac{99}{140}$		$-\frac{37}{210}$
$T_{5,5}$	$\frac{503}{12600}$		$\frac{81}{175}$	$-\frac{1024}{1575}$	$\frac{1053}{1400}$		$-\frac{22}{105}$
$T_{6,5}$	$\frac{1957}{69300}$	$\frac{8192}{17325}$	$-\frac{81}{140}$	$\frac{3904}{5775}$	$-\frac{4779}{7700}$	$\frac{8192}{17325}$	$\frac{683}{6930}$
$T_{6,6}$	$\frac{244963}{8731800}$	$\frac{524288}{1091475}$	$-\frac{729}{1225}$	$\frac{760832}{1091475}$	$-\frac{13851}{21560}$	$\frac{524288}{1091475}$	$\frac{3226}{31185}$

Tabelle 6.1: Gewichte $\bar{\alpha}_j$ zu den Diagonal und Nebendiagonalelementen des Extrapolationstableaus an den Knoten t_j für die Bulirsch–Folge (Beachte $\bar{\alpha}_j = \bar{\alpha}_{n-j}$, $t_j = t_{n-j}$).

Bemerkung 6.2.16 Bei der Lösung von Anfangswertproblemen gewöhnlicher Differentialgleichungen tritt auch die denkbar einfache *harmonische Folge*

$$\mathcal{F}_H = \{1,2,3,\ldots\}, \quad n_i = i$$

auf. In unserem Zusammenhang der Quadratur ist sie jedoch wesentlich ungünstiger als die Romberg-Folge, da zum einen der Aufwand größer ist

$$\mathcal{A}_H = \{2,3,5,7,11,13,19,23,29,\ldots\}$$

zum anderen die Trapezsummen T_{i1} nicht rekursiv berechnet werden können.

Die Berechnung des Extrapolationstableaus wird *zeilenweise* durchgeführt. Es wird abgebrochen, falls genügend viele Ziffern "stehen" oder falls keine Verbesserung der Konvergenz beobachtet wird.

Beispiel 6.2.17 Auf der nächsten Seite findet sich das Extrapolationstableau für das Integral

$$\int_{-1}^{1} \frac{dt}{10^{-4} + x^2}$$

mit der Toleranz eps $= 10^{-8}$.

Extrapolationstableau für $\int_{-1}^{1} \frac{dx}{10^{-4} + x^2}$

exakt: $200 \cdot \operatorname{arctg}(100) =$

$= 312.159332021646 2762..$

					Aufwand
1.9998000200					
10000.9999000100	13333.9999333400				17
5004.4983506447	3338.9978341897	2672.6643609130			
2511.1251266619	1680.0007186677	1569.4009109662	1551.8887927131		33
1274.2036531728	861.8964953431	807.3562137881	795.2602662138	792.2930955217	
674.7708263846	474.9598841219	449.1641100404	443.4785210921	442.0989848367	65
441.7566641029					
409.3025257629	320.8130922224	310.5366394291	308.3362033876	307.8062335143	
307.6749600526	307.6422172677				129
323.6311682677	295.0740491026	293.3581128946	293.0854378703	293.0256309467	
293.0111826548	293.0076017568	293.0067084762			257
312.3613882946	308.6047949703	309.5068446948	309.7631737710	309.8285766569	
309.8450018238	309.8491126465	309.8501406335	309.8503976478		513
312.1593918855	312.0920597491	312.3245440677	312.3692694546	312.3794894376	
312.3819829985	312.3826025299	312.3827571713	312.3827958166	312.3828054770	1025
312.1593307503	312.1593103720	312.1637937468	312.1612421544	312.1604263611	
312.1602122232	312.1601580667	312.1601444889	312.1601410921	312.1601402427	2049
312.1601400304					
312.1593317038	312.1593320216	312.1593334649	312.1592626668	312.1592549041	
312.1592537590	312.1592535250	312.1592534697	312.1592534561	312.1592534528	
312.1592534519	312.1592534517				

(double precision, eps = 10^{-16})

6.3 Adaptive Romberg-Quadratur

6.3.1 Grundidee

Bisher haben wir als Grundschrittweite die Länge des gesamten Intervalls $H = b - a$ zugrunde gelegt. Denken wir an den "Nadelimpuls"

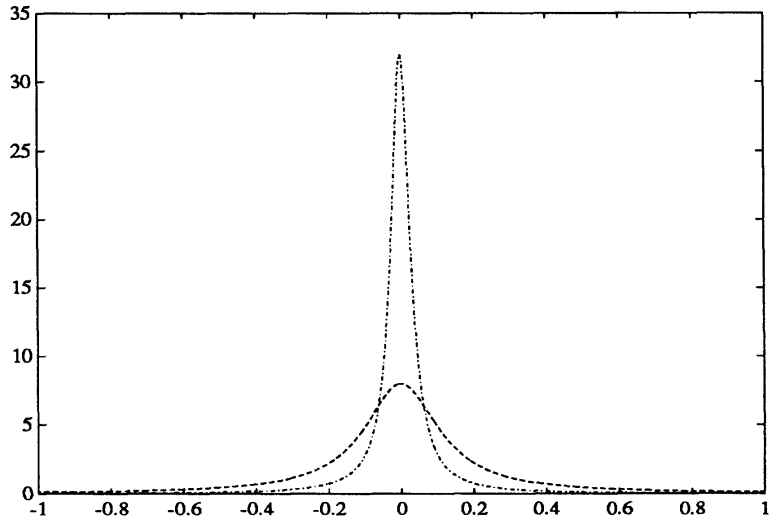


Abbildung 6.25: $f(t) = 2^{-n}/(4^{-n} + t^2)$ für $n = 3$ und $n = 5$

zurück, so sehen wir, daß die entscheidenden Beiträge zum Gesamtintegral häufig nur über einem (oder mehreren) kleinen Teilintervallen liegen. Gehen wir von der Grundschrittweite $H = b - a$ aus, so sind alle Bereiche des Grundintervalls $[a, b]$ gleichberechtigt, wir wenden überall das gleiche Verfahren an. Dies kann nicht der beste Weg sein, um Funktionen wie die in Abbildung 6.25 zu integrieren. Vielmehr sollten wir das Integrationsintervall so unterteilen, daß wir in jedem Bereich ein der Funktion angepaßtes Verfahren wählen und so mit möglichst wenig Aufwand das Integral bis auf eine vorgegebene relative Genauigkeit bestimmen. Derartige Verfahren, die sich selbst steuernd *problemangepaßt* eine Lösung berechnen, nennt man *adaptive Verfahren.* Ihr wesentlicher Vorteil liegt darin, daß eine große Klasse von Problemen mit ein und demselben Programm bearbeitet werden kann, ohne daß der Benutzer Anpassungen vorzunehmen hat, d.h. ohne daß a-priori-Wissen über das Problem in das Verfahren investiert werden muß. Das Programm selbst versucht sich dem Problem anzupassen. Um dies zu erreichen, werden die im Laufe des Algorithmus berechneten Teilergebnisse ständig überprüft. Dies dient zweierlei: Zum einen kann der Algorithmus dadurch automatisch eine bezüglich des Aufwandes optimale Lösungsstrategie auswählen und so die gestellte Aufgabe *effektiv* lösen. Zum anderen wird dadurch gewährleistet, daß das Programm *sicherer* arbeitet und möglichst keine Scheinlösungen abliefert, die in Wirklichkeit herzlich wenig mit dem gestellten Problem zu tun haben. Ziel sollte auch sein, daß das Programm seine eigenen Grenzen erkennt und z.B. angibt, daß eine vorgegebene Genauigkeit nicht erreicht werden kann.

A program may fail, but it must not lie.

In der Quadratur lautet die Aufgabenstellung genauer formuliert:

Aufgabe:

Approximiere das Integral $I = \int_a^b f(t)\,dt$ *auf eine vorgegebene relative Genauigkeit* eps, *d.h. berechne eine Näherung* $\hat{I}$ *von* I, *so daß*

$$|\hat{I} - I| \leq |I| \text{ eps} \tag{6.3.1}$$

Da wir I nicht kennen, ersetzen wir 6.3.1 durch die Forderung

$$|\hat{I} - I| \leq I_{skal} \cdot \text{ eps} \tag{6.3.2}$$

wobei I_{skal} ("skal" wie "Skalierung") in der Größenordnung von $|I|$ liegen soll. Dieser Wert wird entweder vom Benutzer zusammen mit eps vorgegeben oder aus den ersten Approximationen gewonnen.

Während die klassische Romberg-Quadratur lediglich die Ordnung des Verfahrens adaptiert, um eine gewünschte Genauigkeit zu erreichen, wird bei der adaptiven Romberg-Quadratur noch die Grundschrittweite H angepaßt. Dabei gibt es zwei prinzipielle Möglichkeiten, das Problem anzupacken: die Anfangswertmethode (dieser Abschnitt) und die Randwertmethode (übernächster Abschnitt).

Die folgenden Überlegungen stützen sich aus [8] und [7]. Wir gehen aus von der Formulierung des Quadraturproblems als *Anfangswertaufgabe*

$$y'(t) = f(t), \quad y(a) = 0, \quad y(b) = \int_a^b f(t)\,dt$$

und versuchen, das Integral von links nach rechts fortschreitend zu berechnen:

Dabei zerlegen wir das Grundintervall in geeignete, der Funktion f angepaßte Teilintervalle $[t_i, t_{i+1}]$ der Länge $H_i := t_{i+1} - t_i$ und wenden auf die so entstehenden Teilprobleme

$$\int_{t_i}^{t_{i+1}} f(t)\,dt$$

die Romberg-Quadratur bis zu einem bestimmten Grad q_i an.

Bemerkung 6.3.1 Bei diesem Anfangswertzugang wird leider die Symmetrie des Problems

$$I_a^b(f) = -I_b^a(f)$$

zerstört, da wir eine Richtung (von links nach rechts) auszeichnen. Bei dem Randwertzugang (siehe Kapitel 6.5) wird dies nicht der Fall sein.

Diese erste oberflächliche Beschreibung gibt Anlaß zu etlichen Fragen.

a) Welche Schrittweiten soll der Algorithmus wählen ?

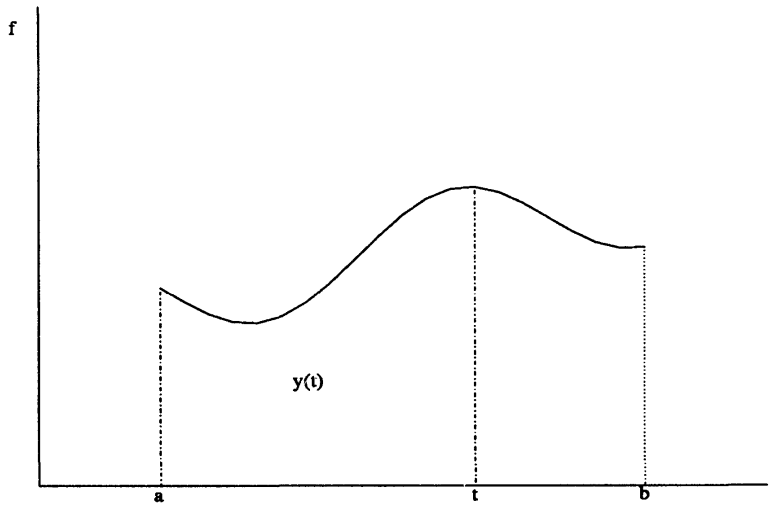
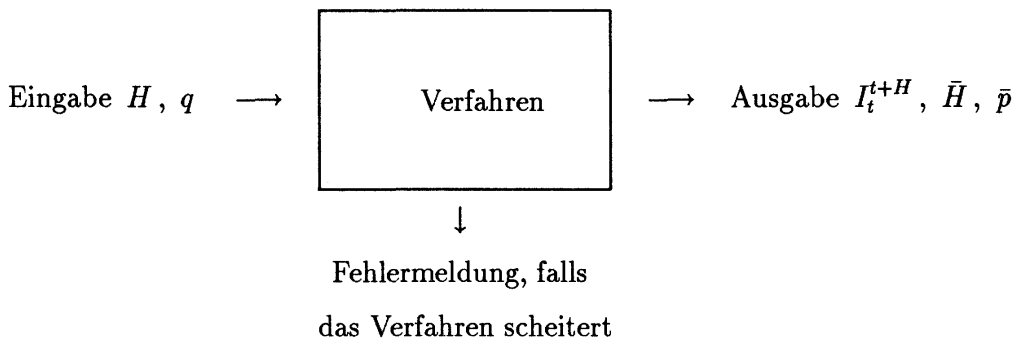


Abbildung 6.26: Quadratur als Anfangswertproblem.

b) Bis zu welcher Ordnung soll die Romberg-Quadratur bezüglich eines Teilintervalls ausgeführt werden ?

c) Wie läßt sich das Ergebnis (lokal) überprüfen ?

Wir konstruieren im folgenden ein Verfahren, welches ausgehend von einer eingegebenen Schrittweite H und einer anfänglich vorgegebenen Ordnung p das Integral des nächsten Teilintervalls berechnet und Vorschläge für die Schrittweite $\bar{H}$ und die Ordnung $\bar{p}$ für den folgenden Schritt macht.



Ist die Berechnung des Teilintegrals $I_t^{t+H}(f)$ auf die gewünschte Genauigkeit mit der vorgegebenen Ordnung nicht möglich, so soll das Verfahren eine neue Ordnung p und/oder eine reduzierte Schrittweite H wählen. Nach "allzu häufiger" Reduktion von H soll das Verfahren abbrechen.

6.3.2 Schätzung des Approximationsfehlers

Sowohl für die Genauigkeitsüberprüfung als auch für die Steuerung des Algorithmus benötigen wir Informationen über die Approximationsgüte unseres Verfahrens. In einem

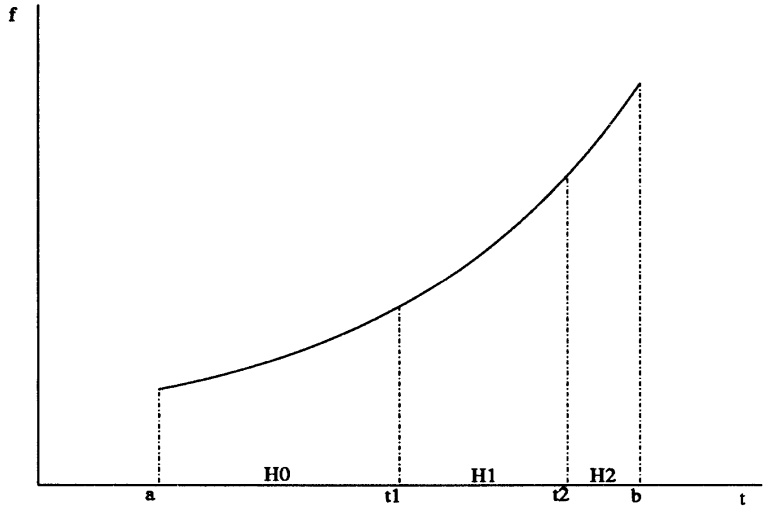


Abbildung 6.27: Zerlegung des Grundintervalls bei der adaptiven Romberg- Quadratur

adaptiven Algorithmus ist es die Aufgabe der sogenannten *Fehlerschätzer*, uns diese Informationen aus bereits berechneten Daten zu verschaffen. Die Bezeichnung "Schätzer" (engl.: estimator) läßt sich wie folgt präzisieren.

Definition 6.3.2 *$\bar{\varepsilon}$ heißt Schätzer für ε ($\bar{\varepsilon} \doteq \varepsilon$), falls es eine Abschätzung nach oben und unten der Form*

$$\kappa_1 \varepsilon \quad \leq \quad \bar{\varepsilon} \quad \leq \quad \kappa_2 \varepsilon$$

mit $\kappa_1 \leq 1 \leq \kappa_2$ gibt.

Die Konstruktion eines Fehlerschätzers gehört zu den schwierigsten Aufgaben bei der Entwicklung eines adaptiven Algorithmus. Eine typische Möglichkeit besteht z.B. darin, eine Approximation niedrigerer Ordnung mit einer höhere Ordnung zu vergleichen. Alle Fehlerschätzer, die wir hier kennenlernen werden, basieren auf diesem Bauprinzip.

Lokal, d.h. bezüglich eines Teilintervalls $[t, t+H]$ mit der Grundschrittweite H, wird in unserem Fall die Approximationsgüte durch das Fehlertableau beschrieben.

$$\begin{array}{llll} \varepsilon_{11} & & & \\ \varepsilon_{21} & \varepsilon_{22} & & \\ \varepsilon_{31} & \varepsilon_{32} & \varepsilon_{33} & \\ \vdots & & & \ddots \end{array}$$

Für die Koeffizienten der asymptotischen Entwicklung der Trapezregel gilt

$$\tau_{2k} \;=\; \frac{B_{2k}}{(2k)!}(f^{(2k-1)}(a+H) - f^{(2k-1)}(a)) \;\doteq\; \underbrace{\frac{B_{2k}}{(2k)!} f^{(2k)}(a)}_{=:\bar{\tau}_{2k}} H \qquad (6.3.3)$$

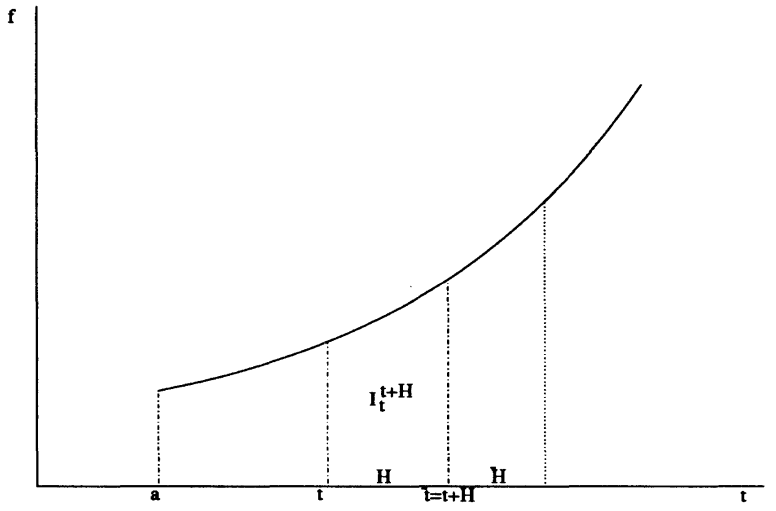


Abbildung 6.28: Ein Schritt der adaptiven Romberg–Quadratur

mit problemabhängigen Konstanten $\bar{\tau}_{2k}$. Eingesetzt in die Formel für den Approximationsfehler ε_{ik} aus Satz 6.2.13 folgt für $1 \leq k \leq i$

$$\begin{aligned} \varepsilon_{ik} &= |T_{ik} - \int_t^{t+H} f(t)\,dt| \\ &\doteq |\bar{\tau}_{2k}|\; h_{i-k+1}^2 \cdots h_i^2 \cdot H \\ &= |\bar{\tau}_{ik}|\; \gamma_{ik}\; H^{2k+1} \quad \text{mit} \quad \gamma_{ik} := (n_{i-k+1} \cdots n_i)^{-2} \end{aligned}$$

Die Ordnung $2k+1$ bzgl. H ist nur abhängig vom Spaltenindex k des Extrapolationstableaus. Insbesondere gilt für zwei aufeinander folgende Fehler in einer Spalte k, unabhängig vom Problem, daß

$$\varepsilon_{i+1,k} \doteq \frac{\gamma_{i+1,k}}{\gamma_{i,k}} \varepsilon_{i,k} = \underbrace{\left(\frac{n_{i-k+1}}{n_{i+1}}\right)^2}_{\ll 1} \varepsilon_{i,k}$$

Mit anderen Worten, unabhängig vom Problem und unabhängig von H werden die Approximationsfehler innerhalb einer Spalte k mit wachsendem Zeilenindex i schnell kleiner. Notieren wir diese Relation im Fehlertableau mit einem Pfeil,

$$\varepsilon \to \delta \quad :\Longleftrightarrow \quad \varepsilon \ll \delta$$

so ergibt sich folgendes Bild:

$$\begin{array}{ccccc} \varepsilon_{11} & & & & \\ \uparrow & & & & \\ \varepsilon_{21} & & \varepsilon_{22} & & \\ \uparrow & & \uparrow & & \\ \varepsilon_{31} & & \varepsilon_{32} & & \varepsilon_{33} \\ \uparrow & & \uparrow & & \uparrow \\ \vdots & & \vdots & & \vdots \end{array}$$

Für die Beziehung zwischen den Spalten benötigt man noch eine weitere Annahme.

Annahme: *Höhere Ordnung ist genauer*, d. h.

$$\varepsilon_{i,k+1} \ll \varepsilon_{ik} \quad \text{für} \quad 1 \leq k < i \tag{6.3.4}$$

Diese Annahme ist zwar plausibel aber nicht zwingend: Sie gilt sicher für "hinreichend kleine" Schrittweiten H, muß aber für konkretes H in geeigneter Weise im Programm überprüft werden. Eine Möglichkeit zu testen, ob unser Modell mit der gegebenen Situation übereinstimmt, werden wir in Kapitel 6.3.4 angeben.

Mit dieser Annahme erhalten wir folgende Situation

$$\begin{array}{ccccccc} \varepsilon_{11} & & & & & & \\ \uparrow & & & & & & \\ \varepsilon_{21} & \leftarrow & \varepsilon_{22} & & & & \\ \uparrow & & \uparrow & & & & \\ \varepsilon_{31} & \leftarrow & \varepsilon_{32} & \leftarrow & \varepsilon_{33} & & \\ \uparrow & & \uparrow & & \uparrow & & \\ \vdots & & \vdots & & \vdots & & \end{array}$$

Die genaueste Approximation innerhalb der Zeile k ist also Diagonalelement T_{kk}. Ideal wäre daher, wenn wir den Fehler

$$\varepsilon_{kk} = |T_{kk} - \int_t^{t+H} f(t)\, dt|$$

abschätzen könnten. An dieser Stelle geraten wir in ein *Dilemma.* Um ε_{kk} abschätzen zu können, benötigen wir eine genauere Approximation $\hat{I}$ des Integrals, also z.B. $T_{k+1,k}$. Mit dieser ließe sich ε_{kk} etwa durch

$$\bar{\varepsilon}_{kk} \;:=\; |T_{k+1,k} - T_{k,k}| \doteq \varepsilon_{kk}$$

abschätzen. Haben wir jedoch $T_{k+1,k}$ berechnet, so können wir auch direkt die (bessere) Approximation $T_{k+1,k+1}$ angeben. Für diese liegt aber keine Abschätzung des Fehlers

$\varepsilon_{k+1,k+1}$ vor, es sei denn, wir berechnen wiederum eine noch genauere Approximation. Aus diesem Dilemma befreit uns nur die Einsicht, daß auch die zweitbeste Möglichkeit nützlich sein kann. Die zweitbeste Lösung, die uns bis zur Zeile k zur Verfügung steht, ist das *subdiagonale Element* $T_{k,k-1}$ und dessen Approximationsfehler $\varepsilon_{k,k-1}$ läßt sich mit bis zu dieser Zeile bekannten Daten wie folgt abschätzen.

Lemma 6.3.3 *Unter der Annahme 6.3.4 ist*

$$\bar{\varepsilon}_{k,k-1} := |T_{k,k-1} - T_{kk}| \doteq \varepsilon_{k,k-1}$$

ein Fehlerschätzer für $\varepsilon_{k,k-1}$.

Beweis: Es ist mit $I := \int_t^{t+H} f(t)\,dt$

$$\begin{aligned} \bar{\varepsilon}_{k,k-1} &= |(T_{k,k-1} - I) - (T_{k,k} - I)| \leq \varepsilon_{k,k-1} + \varepsilon_{kk}\bar{\varepsilon}_{k,k-1} \\ &= |(T_{k,k-1} - I) - (T_{k,k} - I)| \geq \varepsilon_{k,k-1} - \varepsilon_{kk} \end{aligned}$$

und daher

$$(1 - \underbrace{\frac{\varepsilon_{kk}}{\varepsilon_{k,k-1}}}_{\ll 1})\varepsilon_{k,k-1} \leq \bar{\varepsilon}_{k,k-1} \leq (1 + \underbrace{\frac{\varepsilon_{kk}}{\varepsilon_{k,k-1}}}_{\ll 1})\varepsilon_{k,k-1}$$

□

Um die Notation zu erleichtern, gehen wir davon aus, daß $I_{skal} = 1$. Dann ersetzen wir das Abbruchkriterium

$$|I - \hat{I}| \leq \text{eps}$$

durch die Bedingung, daß

$$\bar{\varepsilon}_{k,k-1} \leq \rho\,\text{eps}, \tag{6.3.5}$$

wobei $\rho < 1$ (typischerweise $\rho := 0.25$) ein Sicherheitsfaktor ist. Das Diagonalelement T_{kk} wird also genau dann als Lösung akzeptiert, wenn die Abbruchbedingung 6.3.5 erfüllt ist. Diese Bedingung nennt man auch das *subdiagonale Fehlerkriterium.*

Bemerkung 6.3.4 Es hat lange Zeit heftige Debatten darüber gegeben, ob man die "beste" Lösung (hier: das Diagonalelement T_{kk}) als Näherung ausgeben darf, obwohl nur der Fehler der "zweitbesten" Lösung (hier: des Subdiagonalelements $T_{k,k-1}$) geschätzt wurde. Tatsächlich ist der verwendete Fehlerschätzer für die Lösung $T_{k,k-1}$ nur dann brauchbar, wenn T_{kk} die "beste" Lösung ist, so daß es unsinnig wäre, diese genauere Lösung zu verschenken.

6.3.3 Ordnungs- und Schrittweitensteuerung

Ziel des adaptiven Algorithmus ist es, mit möglichst wenig Aufwand das Integral genau genug zu approximieren. Es wäre von daher ideal, wenn wir die neue Schrittweite $\bar{H}$ so angeben könnten, daß wir das Integral exakt bis auf die vorgegebene Genauigkeit approximieren. Wir gehen zunächst aus von einem Verfahren $\hat{I}$ der gegebenen Ordnung p, d.h.

$$\varepsilon = |\hat{I}_t^{t+H}(f) - \int_t^{t+H} f(\tau)\,d\tau| = \gamma H^{p+1} \tag{6.3.6}$$

mit einer vom linken Rand t und dem Verfahren abhängigen Konstante $\gamma = \gamma(t)$. Wir nehmen an, daß sich diese Konstante in dem Intervall $[t, t+H]$ nur wenig ändert und daher insbesondere am rechten und linken Rand übereinstimmt.

Annahme:

$$\gamma \doteq \bar{\gamma} := \gamma(\bar{t}), \quad \bar{t} := t + H \tag{6.3.7}$$

Für hinreichend kleine H ist diese Annahme immer zu verifizieren.

Ziel: *Bestimme eine Schrittweite $\bar{H}$ so, daß*

$$\bar{\varepsilon} := |\hat{I}_{\bar{t}}^{\bar{t}+\bar{H}}(f) - \int_{\bar{t}}^{\bar{t}+\bar{H}} f(\tau)\, d\tau| = \text{ eps}$$

Mit (6...) gilt

$$\bar{\varepsilon} = \text{ eps} \doteq \gamma \bar{H}^{p+1} = \frac{\varepsilon}{H^{p+1}} \bar{H}^{p+1}$$

so daß wir für $\bar{H}$ die Formel

$$\bar{H} = \sqrt[p+1]{\frac{\text{eps}}{\varepsilon}} \cdot H \tag{6.3.8}$$

erhalten. Wie die Annahme 6.3.4 so muß der Algorithmus auch die Annahme 6.3.7 überprüfen und gegebenenfalls die Schrittweite korrigieren (siehe 6.3.4). Die Formel 6.3.8 läßt sich so interpretieren, daß man die optimale Schrittweite des vorhergehenden Schrittes als neue Schrittweite vorschlägt.

Im vorliegenden Fall der Romberg-Quadratur ist die Ordnung p in der Spalte k des Extrapolationstableaus gerade $p = 2k$ und es gilt

$$\varepsilon_{ik} \doteq |\bar{\tau}_{2k}| \ \gamma_{ik} H^{2k+1} \quad \text{für} \quad f \in C^{2k}[t, t+H]$$

Bisher haben wir noch nicht beachtet, daß wir den Fehler ε überhaupt nicht kennen. Das beste, was wir tun können, ist wie bei der Genauigkeitsabfrage (6...) statt des Diagonalelementes T_{kk} das Subdiagonalelement $T_{k,k-1}$ (welches von der Ordnung $2k-2$ ist) zu betrachten und den Fehler durch $\bar{\varepsilon}_{k,k-1}$ abzuschätzen. Für die Approximation $T_{k,k-1}$ erhalten wir so den Schrittweitenvorschlag

$$\bar{H}_k := \sqrt[2k-1]{\frac{\rho \ \text{eps}}{\bar{\varepsilon}_{k,k-1}}} \cdot H$$

wobei $\rho < 1$ wieder ein Sicherheitsfaktor ist.

Obige Vorschrift liefert nun für jede Spalte k einen Vorschlag $\bar{H}_k$ für die nächste Schrittweite. Es fehlt noch ein Kriterium, um aus diesen Paaren

$$(k, \bar{H}_k) = (\text{Spalte, Schrittweitenvorschlag})$$

das beste auszuwählen. Nun war unser Ziel, den Aufwand zu minimieren; wir könnten auch sagen, mit möglichst wenig Aufwand möglichst weit zu kommen. Das erreichen wir, indem wir die *Arbeit pro Schrittweite* (engl.: *work per unit step*)

$$W_k := \frac{A_k}{\bar{H}_k}$$

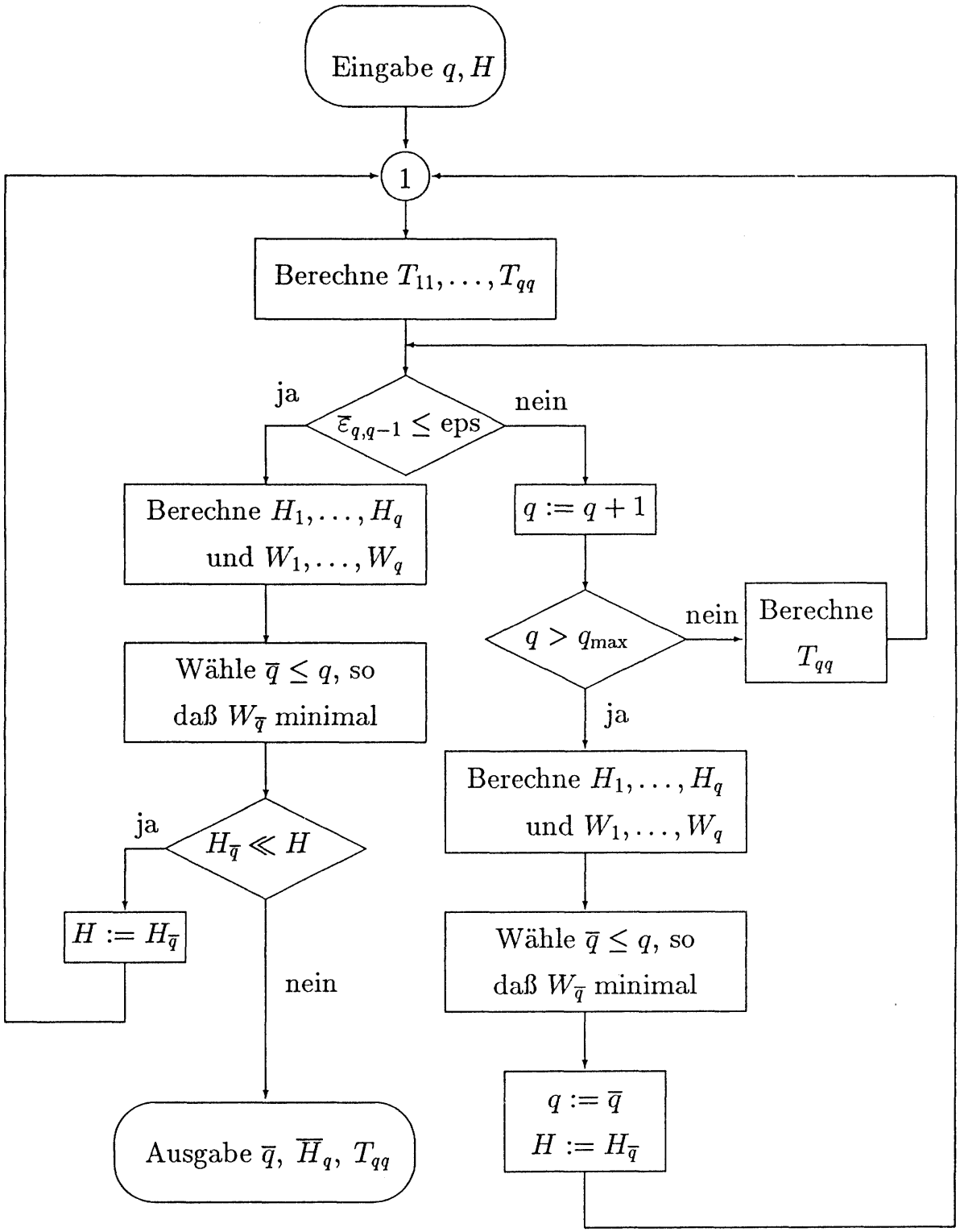
minimieren, wobei A_k die zur Unterteilungsfolge $\mathcal{F}$ gehörenden Aufwandszahlen sind. Die Spalte $\bar{q}$ mit

$$W_{\bar{q}} = \min_{k=1,\ldots,q} W_k$$

ist in diesem Sinn "optimal" ebenso wie die Ordnung $2\bar{q}$.

6.3.4 Herleitung des Algorithmus

Faßt man alle Überlegungen der letzten Abschnitte zusammen, so gelangt man zu folgendem Algorithmus für einen Schritt der adaptiven Romberg-Quadratur.


Eingabe q, H
1
Berechne $T_{11}, \ldots, T_{qq}$
ja
$\overline{\varepsilon}_{q,q-1} \leq \text{eps}$
nein
Berechne $H_1, \ldots, H_q$ und $W_1, \ldots, W_q$
$q := q + 1$
$q > q_{\max}$
nein
Berechne T_{qq}
Wähle $\overline{q} \leq q$, so daß $W_{\overline{q}}$ minimal
ja
Berechne $H_1, \ldots, H_q$ und $W_1, \ldots, W_q$
ja
$H_{\overline{q}} \ll H$
$H := H_{\overline{q}}$
nein
Wähle $\overline{q} \leq q$, so daß $W_{\overline{q}}$ minimal
$q := \overline{q}$
$H := H_{\overline{q}}$
Ausgabe $\overline{q}$, $\overline{H}_q$, T_{qq}

Programmiert man diesen Algorithmus, so muß man auch nach langer Fehlersuche die leidvolle Erfahrung machen, daß er in dieser Form (noch) nicht so funktioniert, wie man dies erwartet hätte. Gerade das Beispiel der Nadelfunktion macht ihm schwer zu schaffen. Zwar ziehen sich die Schrittweiten bei abnehmenden Ordnungen zur Mitte der Nadel hin wie vermutet zusammen, doch bleibt dies auch nach Überschreiten der Nadelspitze erhalten; die Ordnung bleibt niedrig und die Schrittweiten klein.

Zusammen mit zwei weiteren Schwierigkeiten des bisherigen Algorithmus wollen wir diese Situation kurz analysieren.

Nachteile des Algorithmus

1) *Stehenbleiben der Ordnung* (engl.: *trapping of order*), wie oben angesprochen. Hat sich einmal eine niedrige Ordnung q eingestellt und ist die Bedingung $\bar{\varepsilon}_{q,q-1} \leq$ eps immer erfüllt, so testet der Algorithmus keine höhere Ordnung, obwohl diese günstiger sein könnte. Die Ordnung bleibt niedrig und damit die Schrittweiten kurz, wie es bei der Integration der Nadel zu beobachten war.

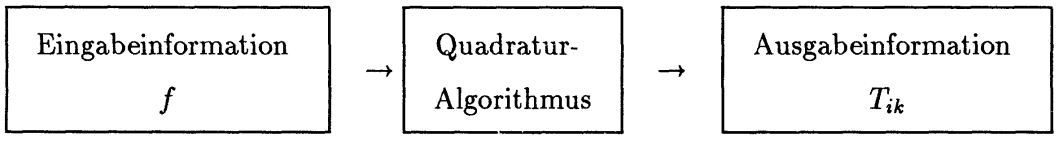
2) Der Algorithmus bemerkt erst sehr spät, nämlich nach Überschreiten von $q_{\max}$, daß eine vorgeschlagene Schrittweite H zu groß war und für keine Spalte q zum Ziel ($\bar{\varepsilon}_{q,q-1} \leq$ eps) führt.

3) Falls unsere Annahmen nicht erfüllt sind, funktioniert der Fehlerschätzer nicht. Daher kann es passieren, daß der Algorithmus eine falsche Lösung als richtig erachtet und ausgibt. Man spricht in diesem Fall von *Pseudokonvergenz.*

Für die beiden letztgenannten Probleme wäre es gut, wenn man bereits frühzeitig erkennen könnte, ob sich die Approximationen "vernünftig", d.h. ganz im Sinne unserer Annahmen, verhalten. Eine solche Möglichkeit nennt man Konvergenz-Monitor (engl.: convergence monitor). Diese Einheit könnte evtl. auch dazu benutzt werden, die Fehler für höhere, noch nicht berechnete Ordnungen hervorzusagen und damit auch das erste Problem zu lösen. Im nächsten Abschnitt wollen wir eine Möglichkeit dazu kurz anreißen.

Um die im letzten Abschnitt angesprochenen Probleme zu lösen und die adaptive Romberg-Quadratur (und allgemeiner adaptive Extrapolationsmethoden) so zu einem äußerst guten Verfahren zu machen, benötigt man ein Modell, das man dem Konvergenzverhalten zugrunde legen und mit dem man die tatsächlich erhaltenen Werte vergleichen kann. Wir wollen hier nur kurz eine solche Möglichkeit diskutieren, welche auf der Shannon'schen Informationstheorie fußt, und verweisen für die Einzelheiten auf [7].

Wir interpretieren den Quadratur-Algorithmus als *Codiermaschine* (engl.: encoding device),



welche die Information, die man durch Auswertung der Funktion erhält, umwandelt in Information über das Integral. Die Informationsmenge auf der Eingabeseite, die *Eingabeentropie* $E_{ik}^{(in)}$, messen wir durch die Anzahl der zur Berechnung von T_{ik} benötigten f-Auswertungen. (Dies setzt voraus, daß keine redundanten f-Auswertungen vorliegen,

d. h. daß alle Ziffern von f unabhängig voneinander sind.) Da zur Berechnung von T_{ik} die Werte $T_{i-k+1,1}, \ldots, T_{i,1}$ als Eingabe benötigt werden, erhalten wir

$$E_{ik}^{(in)} = A_i - A_{i-k} + 1$$

Die Informationsmenge auf der Ausgabeseite, die *Ausgabeentropie* $E_{ik}^{(out)}$, läßt sich durch die Anzahl der gültigen Binärziffern der Approximation T_{ik} charakterisieren. Dies führt auf

$$E_{ik}^{(out)} = \text{ld}\,(\frac{1}{\varepsilon_{ik}})$$

Wir gehen nun davon aus, daß unser Informationskanal mit einem konstanten *Rauschfaktor* $0 < c \leq 1$ arbeitet,

$$E_{ik}^{(out)} = c\, E_{ik}^{(in)} \tag{6.3.9}$$

d.h., daß Ein- und Ausgabeentropie proportional zueinander sind. (Falls $c = 1$, so liegt gerade ein *rauschfreier* Kanal vor; es geht keine Information verloren.) In unserem Fall heißt das

$$\text{ld}\,(\frac{1}{\varepsilon_{ik}}) = c(A_i - A_{i-k} + 1) \tag{6.3.10}$$

Um den Proportionalitätsfaktor c bestimmen zu können, benötigen wir nur ein Paar von Eingabe- und Ausgabeentropie. Wir haben oben verlangt, daß für eine vorgegebene Spalte q der subdiagonale Fehler $\varepsilon_{q,q-1}$ gleich der vorgeschriebenen Genauigkeit eps ist, also

$$\varepsilon_{q,q-1} = \text{eps}$$

Setzen wir diese Beziehung in (.) ein, so folgt

$$-\ \text{ld}\,(\varepsilon_{q,q-1}) = -\ \text{ld}\,(\ \text{eps}) = c(A_q - A_1 + 1) \tag{6.3.11}$$

Haben wir c so bestimmt, so können wir für *alle* i, k angeben, welche Fehler ε_{ik} nach unserem Modell zu erwarten sind. Bezeichnen wir diese Fehler, die das informationstheoretische Modell impliziert mit $\alpha_{ik}^{(q)}$ (wobei q die Zeile ist, mit der wir den Proportionalitätsfaktor gewonnen haben), so folgt

$$\begin{aligned} -\ \text{ld}\,\alpha_{ik}^{(q)} &= c(A_i - A_{i-k} + 1) \\ &= -\ \text{ld}\,(\ \text{eps})\frac{A_i - A_{i-k} + 1}{A_q - A_1 + 1} \end{aligned}$$

und daher

$$\alpha_{ik}^{(q)} = \text{eps}^{\frac{A_i - A_{i-k} + 1}{A_q - A_1 + 1}}$$

Damit haben wir tatsächlich auf recht elementare Weise ein Vergleichsmodell konstruiert, mit dem wir das Konvergenzverhalten unseres Algorithmus gemittelt über eine große Anzahl von Problemen überprüfen können. Dabei vergleichen wir die geschätzten Fehler $\bar{\varepsilon}_{k,k-1}$ mit den Werten $\alpha_{k,k-1}^{(q)}$ des Konvergenzmodells. Wir erhalten so zum einen den gewünschten Konvergenz-Monitor, zum anderen können wir auch abschätzen, wie sich höhere Ordnungen verhalten würden. Dieses Modell wurde in den Algorithmus 6.3.5 eingebaut (Quadratur-Programm TRAPEX).

Ignorieren wir den Sicherheitsfaktor ρ, so approximiert obiger Algorithmus das Integral $I = I(f)$ mit einer *globalen Genauigkeit*

$$|I - \hat{I}| \leq I_{skal} \cdot m \cdot \text{eps},$$

wobei m die Anzahl der Grundschritte ist, die sich bei der adaptiven Quadratur ergeben haben (a-posteriori-Fehlerschätzung). Die gewählte Strategie führt offensichtlich zu einer *Gleichverteilung des lokalen Diskretisierungsfehlers.* Dieses Prinzip ist auch für wesentlich allgemeinere adaptive Diskretisierungsverfahren wichtig — vergleiche auch den nächsten Abschnitt. Falls man einen von m unabhängigen globalen Diskretisierungsfehler

$$|I - \hat{I}| \leq I_{skal} \cdot E,$$

vorschreiben will, so ist in der Herleitung der Ordnungs- und Schrittweitensteuerung die Ersetzung

$$\text{eps} \longrightarrow \frac{H}{b-z} E$$

möglich. Dies führt zu kleineren Änderungen der Ordnungs- und Schrittweitensteuerung.

Bemerkung 6.3.5

1) *Die Wahl von I_{skal} kann äußerst schwierig sein (Skalierungsproblem).*

2) *Die Ordnungs- und Schrittweitensteuerung ist invariant gegenüber linearen Transformationen von t der Form*

$$t \longrightarrow \alpha t, \quad \alpha \neq 0$$

Beispiel 6.3.6 Auf der nächsten Seite findet sich das Ergebnis der adaptiven Romberg-Quadratur (Programm TRAPEX) für das Integral

$$\int_{-1}^{1} \frac{dt}{10^{-4} + x^2}$$

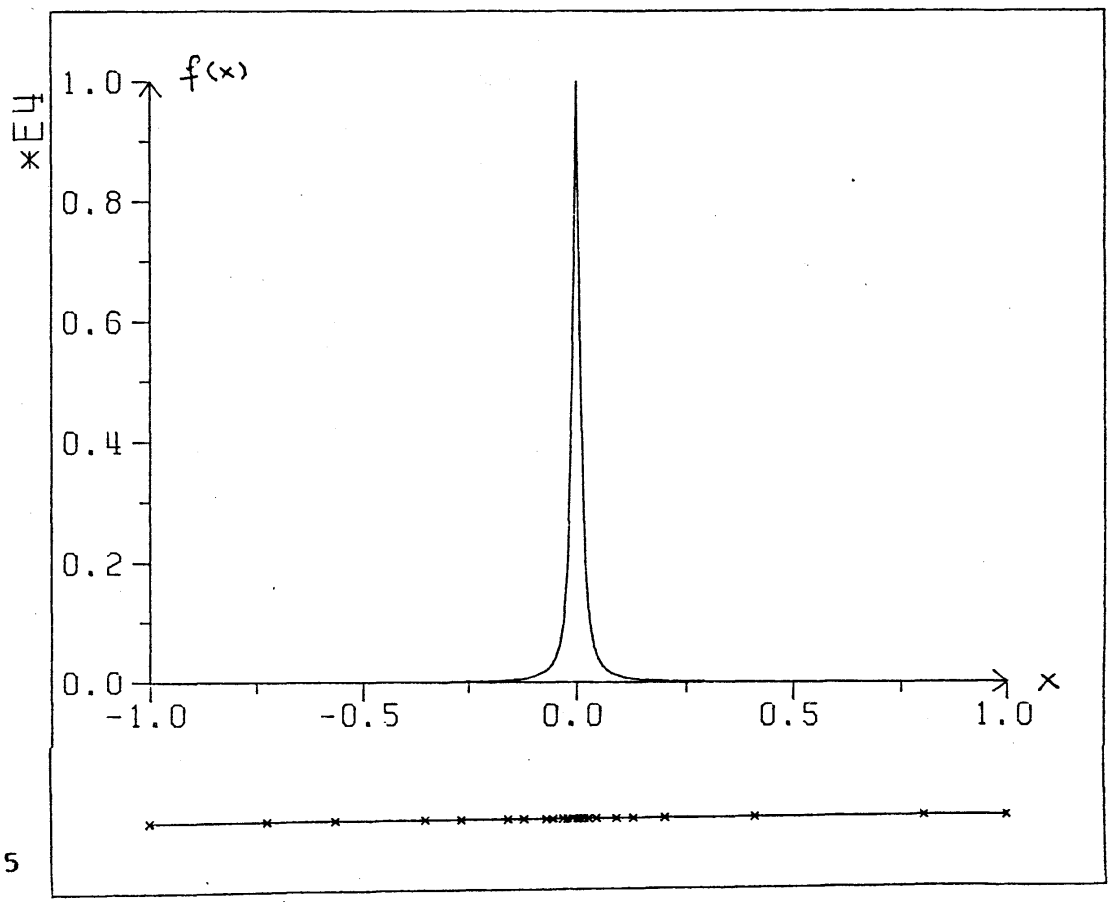
mit der Toleranz $\text{eps} = 10^{-8}$. Es wurden nur 255 f-Auswertungen benötigt.

Integration von $\int_{-1}^{1} f(x)\,dx$

mit Schrittweiten- (u. Ordnungs) steuerung

x_i H_i $TOL = 10^{-8}$ $f(x) = \dfrac{1}{10^{-4} + x^2}$

x_i	H_i
-0.1000000D 01	0.2766393D 00
-0.7233607D 00	0.1612662D 00
-0.5620946D 00	0.2101554D 00
-0.3519392D 00	0.8660897D-01
-0.2653302D 00	0.1103932D 00
-0.1549370D 00	0.3777156D-01
-0.1171654D 00	0.5058397D-01
-0.6658143D-01	0.1477522D-01
-0.5180621D-01	0.2344187D-01
-0.2836434D-01	0.9448428D-02
-0.1891591D-01	0.1193243D-01
-0.6983486D-02	0.9181705D-02
0.2198220D-02	0.4256733D-02
0.6454953D-02	0.5769718D-02
0.1222467D-01	0.7328300D-02
0.1955297D-01	0.1017122D-01
0.2972419D-01	0.1758839D-01
0.4731259D-01	0.4655420D-01
0.9386679D-01	0.3846323D-01
0.1323300D 00	0.7233035D-01
0.2046604D 00	0.2068257D 00
0.4114861D 00	0.3957510D 00
0.8072372D 00	0.1927628D 00



ERGEBNIS: 0.3121593328969358D 03

NSTEP: 23 IFCN: 255

6.4 Spezielle Probleme der numerischen Quadratur

Natürlich kann auch die adaptive Romberg-Quadratur nicht alle Probleme der numerischen Quadratur lösen. In diesem Kapitel wollen wir fünf Schwierigkeiten und deren Behandlung diskutieren.

1. Unstetigkeiten des Integranden

Ein häufiges Problem der numerischen Quadratur sind Unstetigkeiten des Integranden f oder seiner Ableitungen

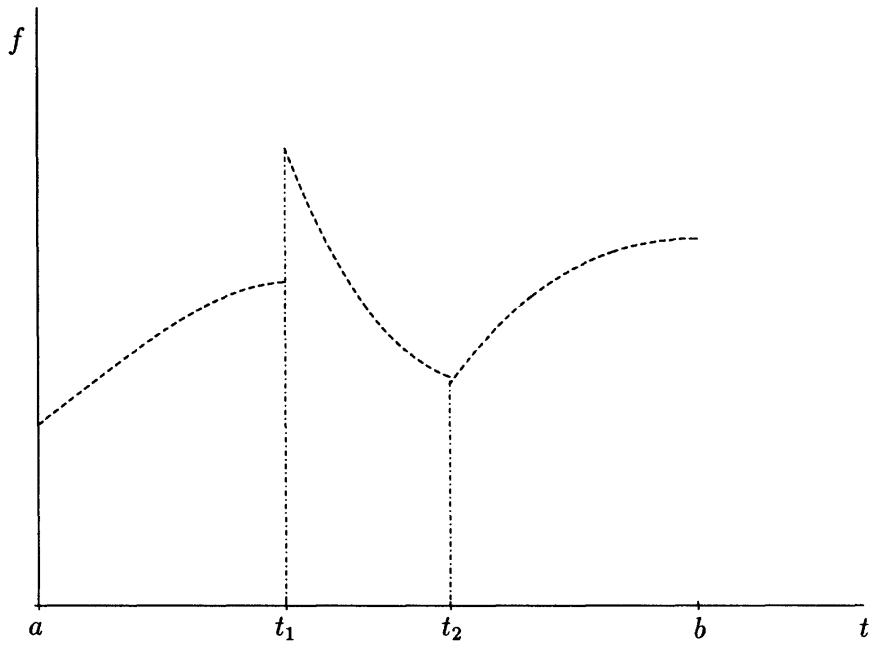


Abbildung 6.29: Sprungstelle von f bei t_1, Sprungstelle von f' bei t_2

Solche Integranden tauchen z.B. auf, wenn ein physikalisch-technisches System in verschiedenen Bereichen mit unterschiedlichen Modellen beschrieben wird, die an den Nahtstellen nicht ganz zusammenpassen. Sind die Sprungstellen bekannt, so sollte man das Integrationsintervall an diesen Stellen zerlegen und die so entstehenden Teilprobleme separat lösen. Im anderen Fall reagieren die Quadraturprogramme sehr unterschiedlich

a) nicht-adaptive Quadratur: Ohne Vorbehandlung liefert ein nicht-adaptives Quadraturprogramm falsche Ergebnisse oder konvergiert nicht. Die Sprungstellen können nicht lokalisiert werden

b) adaptive Romberg-Quadratur: Das adaptive Romberg-Verfahren frißt sich an Sprungstellen fest. Die Sprungstellen können so lokalisiert und extra behandelt werden.

2. Schmale Peaks

Mit diesem Problem haben wir uns in diesem Kapitel hinreichend beschäftigt. Jedes Quadratur-Programm versagt, wenn die Spitzen nur schmal genug sind. Andererseits sind solche Integranden recht häufig, man denke z.B. an das Spektrum eines Sterns, dessen Gesamtstrahlung berechnet werden soll. Ist bekannt, an welchen Stellen die Spitzen liegen, so sollte man das Intervall geeignet unterteilen und wieder die Teilintegrale einzeln berechnen. Sonst bleibt nur die Hoffnung, daß das adaptive Quadratur-Programm sie nicht übersieht.

3. Schwach singuläre Integranden

Definition 6.4.1 *Eine Funktion* $f \in C[a,b]$ *heißt* schwach singulär *in* $[a,b]$, *falls eine ihrer Ableitungen* $f^{(k)}$ *in* $[a,b]$ *eine Singularität besitzt.*

Beispiel 6.4.2

$$\int_{t=0}^{\pi} \underbrace{\sqrt{t}\cos\ t}_{f(t)}\, dt$$

Die Ableitung $f(t) = \frac{\cos\ t}{2\sqrt{t}} - \sqrt{t}\sin\ t$ *hat bei* 0 *einen Pol* $\lim_{t\to 0} f(t) = \infty$.

Integrale mit schwachen Singularitäten machen Quadraturprogramme sehr langsam. Häufig können die Singularitäten durch Substitution beseitigt werden.

Beispiel 6.4.3

$$s = \sqrt{t} \quad \int_{t=0}^{\pi} \sqrt{t}\ \cos\ t\ dt = 2\int_{s=0}^{\sqrt{\pi}} s\ \cos\cdot s^2 ds$$

Dies wird allerdings ineffizient, wenn die Substitution zu schwer auszuwertenden Funktionen führt (z.B. t^α statt $\sqrt{t}$ für eine $0 < \alpha < 1$). Eine zweite Möglichkeit besteht in der rekursiven Auflösung des zu berechnenden Integrals (Miller-Trick), auf die wir hier nicht eingehen wollen.

4. Quadratur bei parameterabhängigen Problemen

Oft hängt der Integrand f von einem zusätzlichen Parameter $p \in \mathbf{R}$ ab

$$f(t,p), \quad p \in \mathbf{R} \text{ Parameter}$$

Wir haben somit eine ganze Schar von Problemen

$$I(p) := \int_a^b f(t,p)dt$$

zu lösen. Die wichtigste Beispielklasse für solche parameterabhängigen Integrale ist die mehrdimensionale Quadratur. Zumeist ist der Integrand nach p differenzierbar und damit auch das Integral $I(p)$. Natürlich wünscht man sich, daß die Approximation $\hat{I}(p)$ diese Eigenschaft erbt. Leider stellt es sich heraus, daß gerade unsere besten Methoden, die adaptiven Quadratur-Verfahren, diese Eigenschaft nicht haben im Gegensatz zu den einfachen nicht-adaptiven Quadratur-Formeln. Es gibt im wesentlichen drei Möglichkeiten, den adaptiven Ansatz bei parameterabhängigen Problemen zu retten.

a) *Einfrieren der Ordnungen und Schrittweiten*

Man führt die Quadratur für einen Parameterwert aus, speichert dabei die benutzten Ordnungen und Schrittweiten ab und verwendet für alle weiteren Parameterwerte dieselbe Ordnungs- und Schrittweitenfolge.

Dies kann nur gutgehen, wenn sich der Integrand in Abhängigkeit des Parameters qualitativ nicht stark verändert. Wandert jedoch z.B. eine Spitze mit dem Parameter und ist diese Abhängigkeit bekannt, so hilft die zweite Möglichkeit.

b) *Parameterabhängige Gitter*

Man transformiert das Integral in Abhängigkeit von p derart, daß der Integrand qualitativ stehenbleibt (die Wanderung der Spitze wird z.B. gerade wieder rückgängig gemacht) oder verschiebt in Abhängigkeit von p die adaptive Zerlegung des Integrationsintervalls aus a).

Die letzte Möglichkeit erfordert viel Einsicht in das jeweilige Problem.

c) *Feste Quadraturformel über sinnvollem Gitter*

Man wählt für das jeweilige Problem ein festes, dem Problem angepaßtes Gitter und integriert über diesem Gitter mit einer festen Quadraturformel (Newton-Cotes ect.). Dazu müssen die qualitativen Eigenschaften des Integranden natürlich weitgehend bekannt sein.

5. Diskrete Integranden

In vielen Anwendungen liegt der Integrand nicht als Funktion f, sondern nur in Form von endlich vielen diskreten Punkten

$$(t_i, f_i) \quad i = 0, \ldots, N$$

vor (z.B. Kernspektrum, digitalisierte Meßdaten). Die einfachste und beste Möglichkeit, damit umzugehen, besteht darin, die Trapezsumme über diese Punkte zu bilden. Die Trapezsumme hat den Vorteil, daß sich Meßfehler bei der Berechnung des Integrals bei äquidistantem Gitter häufig ausmitteln. Falls die Meßfehler δf_i den Erwartungswert 0 haben, d.h. $\sum_{i=0}^{N} \delta f_i = 0$, so gilt dies auch für den dadurch verursachten Fehler der Trapezsumme. Diese Eigenschaft geht bei Verfahren höherer Ordnung verloren.

6.5 Adaptive Multilevel-Quadratur

In diesem Abschnitt befassen wir uns mit einem zweiten Zugang zur adaptiven Quadratur, der auf Ideen beruht, die ursprünglich für die Lösung von Randwertproblemen partieller Differentialgleichungen entwickelt und von M. Wulkow [20] auf das Quadraturproblem übertragen wurden. Dieser sogenannte *Multilevel-Zugang* (deutsch: *Mehrebenen-Zugang*) unterscheidet sich von der adaptiven Romberg-Quadratur durch seinen *globalen* oder *Randwert-* Ansatz. Bei der adaptiven Romberg-Quadratur haben wir das Intervall

in einer willkürlich gewählten Richtung durchlaufen, dabei problemangepaßt in Teilintervalle unterteilt und über diesen mit *lokalen Feingittern* (der Romberg-Quadratur) integriert. Im Gegensatz dazu geht die Multilevel-Quadratur vom gesamten Grundintervall oder einer groben Ausgangsunterteilung aus und erzeugt Schritt für Schritt eine Folge von feineren Unterteilungen Δ^i des Intervalls und genaueren Approximationen $I(\Delta^i)$ des Integrals. Dabei werden die Gitter nur dort verfeinert, wo es für die geforderte Genauigkeit notwendig ist, d.h. das qualitative Verhalten des Integranden wird bei der Verfeinerung der Gitter sichtbar.

Beispiel 6.5.1

$$I(f) = \int_{-1}^{1} f(t)\,dt \quad \text{für} \quad f(t) = \frac{1}{10^{-4} + x^2}$$

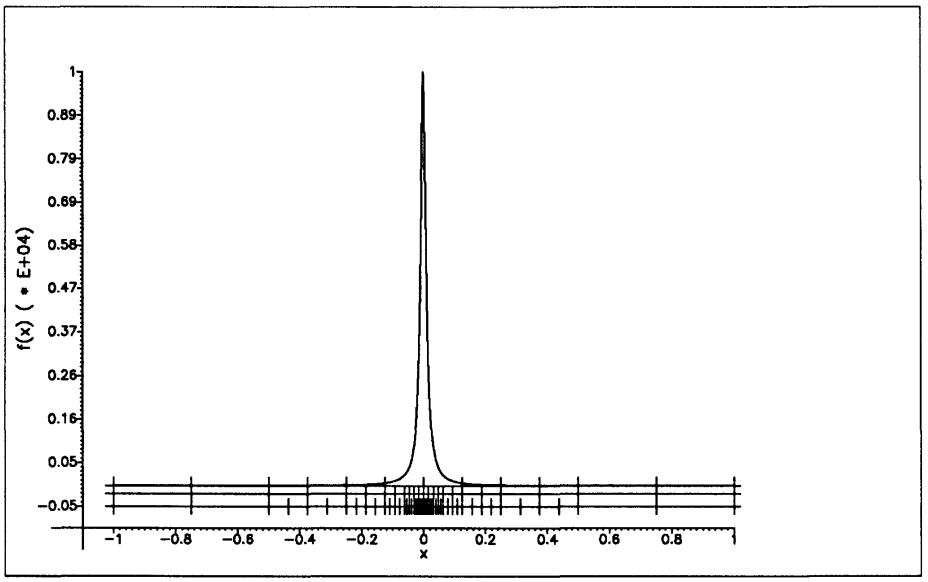


Abbildung 6.30: Gitter der Stufen 3, 6, und 9 für die Toleranz 10^{-3}.

Die Knoten sammeln sich dort, wo "viel passiert". Um dies zu erreichen, benötigt man (wie bei jedem Multilevel-Verfahren) zweierlei

a) einen *lokalen Fehlerschätzer*

b) *Verfeinerungsregeln*

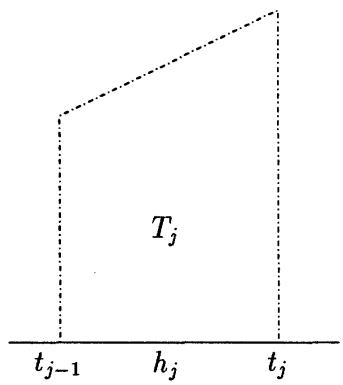
Der lokale Fehlerschätzer wird typischerweise durch den Vergleich von Verfahren niedrigerer und höherer Ordnung realisiert. In diesem ersten Teil der Konstruktion eines Multilevel-Verfahrens müssen Methoden der Analysis angewandt werden. Im zweiten Teil, der Aufstellung von Verfeinerungsregeln, spielen Aspekte der *Datenstrukturen* die entscheidende Rolle. Tatsächlich kann so ein Teil der Komplexität des mathematischen Problems auf die Seite der Informatik (in Form komplexerer Datenstrukturen) verlagert werden.

Für die Multilevel-Quadratur verwenden wir die Trapezregel als Verfahren niedrigerer und die Simpson'sche Regel als Verfahren höherer Ordnung

a) *Trapezregel* (lokal linear)

b) *Simpson-Regel* (lokal quadratisch)

Damit haben wir die elementaren Bausteine für den lokalen Fehlerschätzer zusammengetragen.



$$T_j := \frac{h_j}{2}(f(t_{j-1}) + f(t_j))$$

$$h_j := t_j - t_{j-1}$$

Abbildung 6.31: Trapezregel

6.5.1 Lokaler Fehlerschätzer

Wir führen folgende Bezeichnungen für das i-te Gitter Δ^i und die Teilintervalle der Stufe i ein:

$$\begin{aligned} \Delta^i &= \{t_0^{(i)}, \ldots, t_{n_i}^{(i)}\} \text{ für } i = 0, 1, \ldots \\ I_j^i &= [t_{j-1}^{(i)}, t_j^{(i)}] \text{ für } j = 1, \ldots, n_i \\ n_i &= \text{Anzahl der Teilintervalle auf Stufe } i \end{aligned}$$

Analog seien T_j^i und S_j^i die Approximationen von $\int_{I_j^i} f(t)dt$ mit der Trapez- bzw. Simpson-Regel. Unser Ziel ist es, einen Schätzer für den lokalen Approximationsfehler der Trapezregel über einem Intervall I_j^i

$$\varepsilon_j^i := |T_j^i - \int_{I_j^i} f(t)dt|$$

$$S_j = \frac{h_j\left[(2h_j + 3h_{j+1})h_{j+1}f(t_{j-1}) + (h_j + 3h_{j+1})(h_j + h_{j+1})f(t_j) - h_j^2 f(t_{j+1})\right]}{6h_{j+1}(h_j + h_{j+1})}$$

$$S_{j+1} = \frac{h_{j+1}\left[(2h_{j+1} + 3h_j)h_j f(t_{j+1}) + (h_{j+1} + 3h_j)(h_{j+1} + h_j)f(t_j) - h_{j+1}^2 f(t_{j-1})\right]}{6h_j(h_j + h_{j+1})}$$

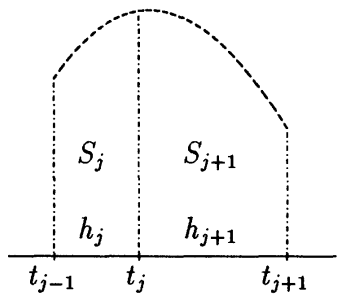


Abbildung 6.32: Simpson–Regel

zu konstruieren. Wie bei der Romberg-Quadratur gehen wir davon aus, daß das Verfahren höherer Ordnung, die Simpson-Regel, lokal besser ist, d.h.

$$|T_j^i - \int_{I_j^i} f(t)dt| \gg |S_j^i - \int_{I_j^i} f(t)dt| \tag{6.5.1}$$

und setzen dann

$$\bar{\varepsilon}_j^i := |T_j^i - S_j^i|$$

Unter obiger Annahme ist dies ein Schätzer für den Fehler ε_j^i, d. h.

$$\bar{\varepsilon}_j^i \doteq \varepsilon_j^i$$

und das Simpson-Resultat als genauere Approximation verwendbar.

6.5.2 Verfeinerungsregeln

Wir betrachten ein Intervall I_k^{i-1} zusammen mit seinen möglichen Verfeinerungen über zwei Stufen, wobei wir uns auf die *Zweiteilung* (*Bisektion*) eines Intervalls als mögliche Verfeinerungsmethode beschränken .

Das Prinzip, nach dem wir bei der Aufstellung der Verfeinerungsregeln vorgehen wollen, ist die *Gleichverteilung des lokalen Diskretisierungsfehlers*, vgl. Kapitel 6.3.4.

Verfeinerungsziel:

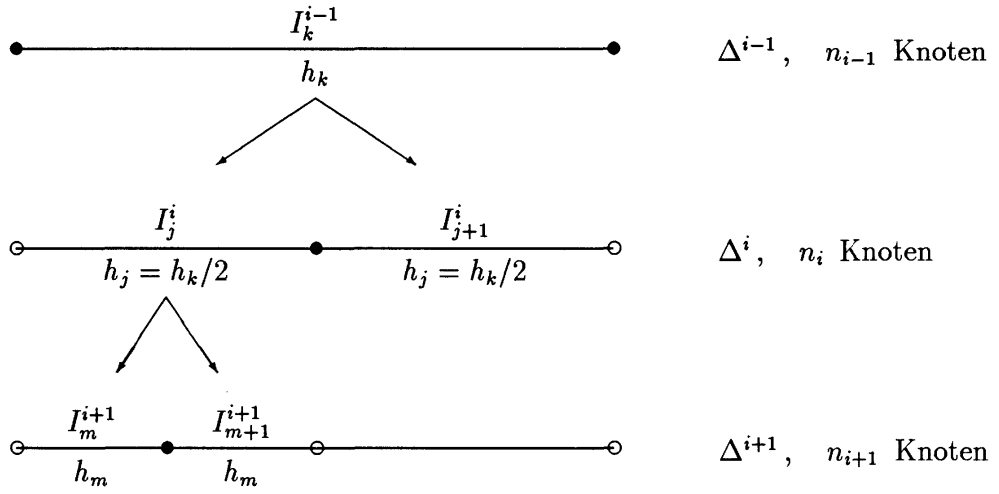


Abbildung 6.33: Bisektion eines Intervalls über zwei Stufen

Verfeinere das Gitter Δ^i so, daß die lokalen Approximationsfehler ε_j^{i+1} des verfeinerten Gitters Δ^{i+1} möglichst gleich sind, d.h.

$$\varepsilon_j^{i+1} \approx \varepsilon_k^{i+1} \quad \textit{für} \quad j,k = 1,\ldots,n_i \tag{6.5.2}$$

Da wir die Fehler ε_j^{i+1} nicht kennen, fordern wir anstelle von 6.5.2 für die Fehlerschätzungen $\bar{\varepsilon}_j^{i+1}$

$$\bar{\varepsilon}_j^{i+1} \approx \bar{\varepsilon}_k^{i+1} \quad \text{für} \quad j,k = 1,\ldots,n_i \tag{6.5.3}$$

Wie wir wissen, verhält sich der Fehler (und damit unter der Voraussetzung 6.5.1 auch sein Schätzer) nach der Formel

$$\bar{\varepsilon}_j^i \doteq C h_j^\gamma, \quad C \text{ lokal konstant}, \gamma \text{ Ordnung} \tag{6.5.4}$$

Bemerkung: Die Trapezregel hat eigentlich die Ordnung $\gamma = 2$. In der Konstante versteckt sich jedoch die zweite Ableitung des Integranden, so daß eine Ordnung $\gamma < 2$ das Verfahren realistischer charakterisiert, wenn wir davon ausgehen, daß C lokal konstant ist. In den folgenden Überlegungen kürzt sich die Ordnung γ heraus, so daß wir damit keine Probleme bekommen.

Damit können wir einen zweiten Fehlerschätzer $\tilde{\varepsilon}_m^{i+1}$ definieren, der uns Information über den Fehler ε_m^{i+1} der nächsten Stufe liefert für den Fall, daß wir das Intervall I_j^i unterteilen. Es ist

$$\begin{aligned} \bar{\varepsilon}_k^{i-1} &\doteq C h_k^\gamma \\ \bar{\varepsilon}_j^i &\doteq C(\frac{h_k}{2})^\gamma = C h_k^\gamma 2^{-\gamma} \doteq 2^{-\gamma} \bar{\varepsilon}_k^{i-1} \end{aligned}$$

und daher

$$\bar{\varepsilon}_m^{i+1} \doteq C h_m^\gamma = C h_k^\gamma 2^{-2\gamma} \doteq \bar{\varepsilon}_k^{i-1} \left(\frac{\bar{\varepsilon}_j^i}{\bar{\varepsilon}_k^{i-1}} \right)^2$$

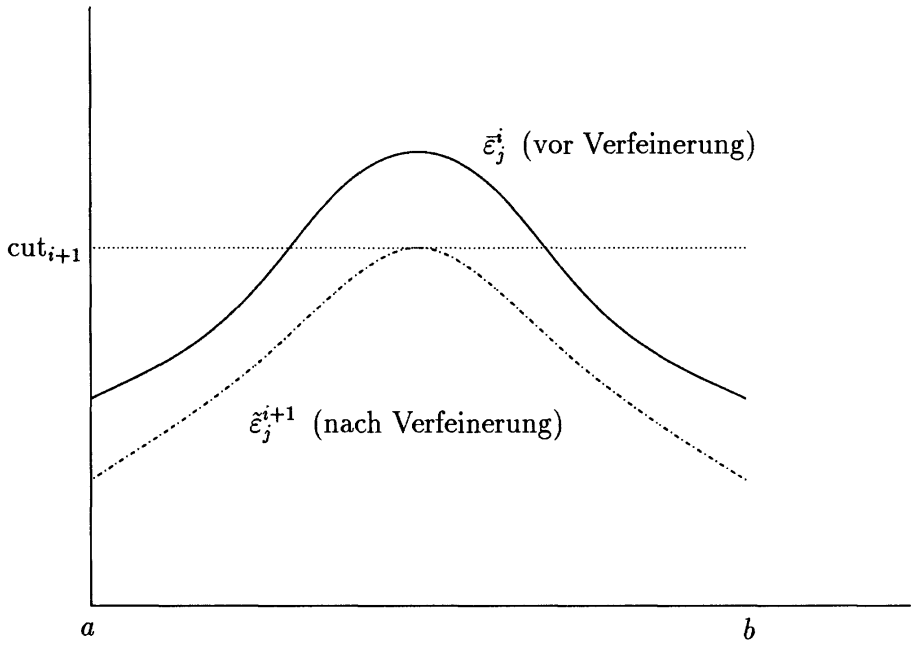


Abbildung 6.34: Fehlerverteilung vor und nach kompletter Verfeinerung

Also ist

$$\tilde{\varepsilon}_m^{i+1} := \frac{(\bar{\varepsilon}_j^i)^2}{\bar{\varepsilon}_k^{i-1}}$$

ein Schätzer für den Fehler ε_m^{i+1}, da

$$\tilde{\varepsilon}_m^{i+1} \doteq \varepsilon_m^{i+1}$$

Damit können wir auf Stufe i abschätzen, wie sich eine Verfeinerung eines Intervalls I_j^i auswirken würde. Wir müssen nun nur noch eine Grenze für die lokalen Fehler festlegen, oberhalb derer wir ein Intervall verfeinern. Dazu ziehen wir den maximalen lokalen Fehler heran, den wir bei Verfeinerung aller Intervalle von Stufe i erhalten würden und definieren

$$\text{cut}_{i+1} := \max_{j=1,\ldots,n_i} \tilde{\varepsilon}_j^{i+1}$$

Zur Veranschaulichung der Situation tragen wir die geschätzten Fehler $\bar{\varepsilon}_j^i$ und $\tilde{\varepsilon}_j^{i+1}$ in einem Histogramm auf

$\bar{\varepsilon}_j^i$: Fehlerschätzer für Stufe i mit Informationen von Stufe i

$\tilde{\varepsilon}_j^{i+1}$: Fehlerschätzer für Stufe $i+1$ mit Informationen von den Stufen i und $i-1$

Am rechten und linken Rand liegt der Fehler bereits vor der Verfeinerung unter dem bei vollständiger Verfeinerung bestenfalls erreichbaren lokalen Fehler cut_{i+1}. Folgen

wir dem Prinzip der Gleichverteilung des lokalen Fehlers, so brauchen wir dort nicht zu verfeinern. Nur im mittleren Bereich zahlt sich die Unterteilung aus. Wir kommen so zu der Verfeinerungsregel:

Verfeinerungsregel:

Verfeinere nur solche Intervalle, für die gilt:

$$\bar{\varepsilon}_j^i \geq \mathrm{cut}_{i+1}$$

Damit ergibt sich in obigem Histogramm in etwa die folgende Fehlerverteilung.

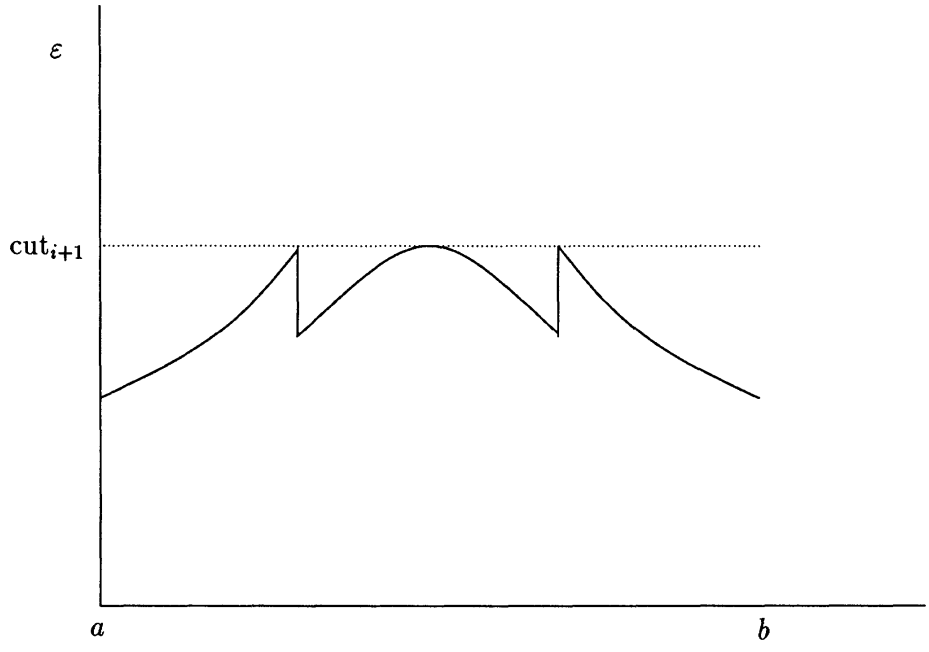


Abbildung 6.35: Fehlerverteilung nach Verfeinerung

Den *globalen Approximationsfehler* können wir durch

$$\bar{\varepsilon}_{total}^i := \sum_{i=0}^{n_i} \bar{\varepsilon}_j^i$$

abschätzen. Bei lokaler Verfeinerung geht der Anteil des Intervalls I_j^i über in die beiden Anteile der entstehenden Teilintervalle.

$$\bar{\varepsilon}_j^i \to \bar{\varepsilon}_m^{i+1} + \bar{\varepsilon}_{m+1}^{i+1} \doteq 2\bar{\varepsilon}_j^{i+1} \doteq 2^{1-\gamma}\bar{\varepsilon}_j^i$$

Damit die Unterteilung also tatsächlich eine Verbesserung erbringt, muß lokal für die Ordnung γ die Bedingung

$$\gamma > 1$$

erfüllt sein.

6.5.3 Details des Algorithmus

Der vollständige Algorithmus der adaptiven Multilevel-Quadratur sieht nun wie folgt aus:

1. (Startphase) Wähle ein Startgitter Δ^0, das die Randpunkte a, b enthält, $\{a, b\} \subset \Delta^0$, und aus mindestens zwei Teilintervallen besteht, z.B. $\Delta^0 = \{a, (a+b)/2, b\}$. (Die letzte Bedingung wird für die Simpsonregel verwendet). Setze $i := 0$.

2. Berechne alle Teilintegrale T_j^i, S_j^i auf der Stufe i und daraus die lokalen Fehlerschätzer $\tilde{\varepsilon}_j^i$ und den globalen Fehlerschätzer $\tilde{\varepsilon}_{total}^i$.

3. Ist die globale Genauigkeit erreicht, d.h. gilt

$$\tilde{\varepsilon}_{total}^i \leq I_{skal} \cdot eps$$

   so breche den Algorithmus ab mit der Summe der Simpson-Approximation als Lösung.

4. Berechne die lokalen Fehlerschätzer $\tilde{\varepsilon}_j^{i+1}$ für die nächste Stufe $i+1$ und verfeinere alle Intervalle I_j^i, deren geschätzter Fehler über der Abschneidegrenze cut_{i+1} liegt. Gehe über zur nächsten Stufe, $i \rightarrow i+1$, und wiederhole den Vorgang ab 2.

Offensichtlich führt der Multilevel-Zugang zu einem wesentlich einfacheren adaptiven Quadratur-Algorithmus als dies bei der adaptiven Romberg-Quadratur der Fall war. Die einzige Schwierigkeit (die bei Verwendung einer strukturierten Programmiersprache wie *C* oder Pascal/Modula eigentlich gar keine mehr ist) besteht in der Speicherung der Gitterfolge. Bei der eindimensionalen Quadratur läßt sich diese Folge als Binärbaum abspeichern (wie in Abbildung 6.33 angedeutet). Bei anderen Problemen birgt die Frage der Datenstrukturen oft eine weit höhere Komplexität in sich, man denke nur an die Verfeinerung von Tetraedernetzten.

Bemerkungen:

1) Das Programm läßt sich auch an diskrete Integranden anpassen (ursprünglich wurde es sogar von M.Wulkow zu dem Zweck als sogenannter *Summator* entwickelt). Dazu muß man sich nur eine Strategie überlegen für den Fall, daß für einen Halbierungspunkt kein Wert vorliegt. Wie immer tun wir das nächstliegendste, diesmal sogar im wörtlichen Sinn, indem wir statt des Halbierungspunktes den nächstliegenden gegebenen Punkt nehmen und so die Bisektion leicht modifizieren. Ist die gewünschte Genauigkeit erreicht, so nehmen wir bei diskreten Integranden aus den oben diskutierten Gründen die *Trapezsumme* als beste Näherung. Als Beispiel zeigen wir die Summation der harmonischen Reihe. Statt 10^7 Termen wurden bei einer Genauigkeit von $\text{eps} = 10^{-3}$ nur ca. 150 Terme benötigt.

2) Das Multilevel-Konzept überträgt sich auf wesentlich allgemeinere Randwertaufgaben und hat sich bisher stets als einfach (bis auf die Datenstrukturen), schnell und zuverlässig erwiesen.

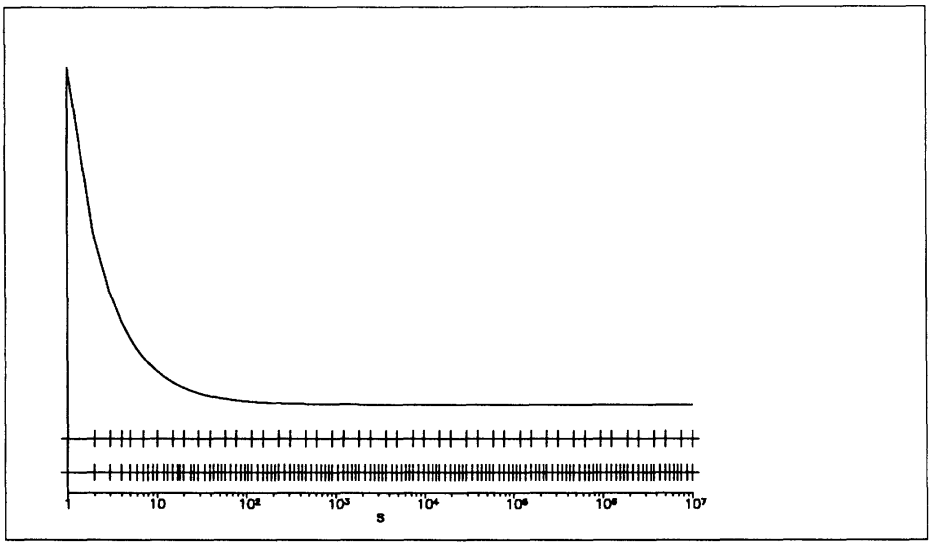


Abbildung 6.36: Summation der harmonischen Reihe mit dem Programm SUMMATOR.

6.6 Übungen

Aufgabe 6.1 Zeigen Sie für die Konstanten $\bar{\alpha}_{in}$ der Newton-Cotes-Formeln

$$\bar{\alpha}_{in} = \frac{1}{n}\int_0^n \frac{(s-0)(s-1)\cdots(s-(i-1))(s-(i+1))\cdots(s-n)}{(i-0)(i-1)\cdots(i-(i-1))(i-(i+1))\cdots(i-n)}dt$$

, die fogenden Gleichungen

a) $\bar{\alpha}_{n-i,n} = \bar{\alpha}_{in}$

a) $\sum_{i=0}^{n} \bar{\alpha}_{in} = 1$

Aufgabe 6.2 Leiten Sie die Formel

$$T_{ik} = T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{\left(\frac{n_i}{n_{i-k+1}}\right)^2 - 1}$$

für das Extrapolationstableau aus der des Aitken-Neville-Algorithmus ab.

Aufgabe 6.3 Zeigen Sie, daß die klassische Romberg-Quadratur mit der Romberg-Folge

$$F_R = \{1, 2, 4, 8, 16, \ldots\}, \quad \text{d. h. } n_i = 2^{i-1}$$

zu Quadraturformeln mit positiven Gewichten α_{in} führt.

Aufgabe 6.4 Einige der durch die Romberg-Quadratur erzeugten Quadraturformeln entsprechen gewissen Newton–Cotes–Verfahren. Zeigen Sie:

a) Mit $h_1 = b - a$ und $h_2 = (b-a)/2$ entspricht T_{22} der Simpson–Regel.

b) Mit $h_1 = b-a$, $h_2 = (b-a)/2$ und $h_3 = (b-a)/4$ entspricht T_{33} der Milne–Regel.

Aufgabe 6.5 Versuchen Sie sich an einem adaptiven Romberg-Quadratur-Programm, testen Sie dieses mit der "Nadelfunktion"

$$I(n) := \int_{-1}^{1} \frac{2^{-n}}{4^{-n} + t^2} dt, \quad \text{für } n = 1, 2, \ldots$$

und geben Sie dasjenige n an, für das Ihr Programm bei einer vorgegebenen Genauigkeit von $eps = 10^{-3}$ den Wert Null liefert.

Literatur

[1] Wilkinson, *Rundungsfehler*, Springer-Verlag Berlin, Heidelberg, New York

[2] Wilkinson, *The Algebraic Eigenvalue Problem*

[3] Stewart, *Introduction to Matrix Computations*

[4] Nicholas J. Higham, *How accurate is Gaussian Elimination?*, Num. Anal. Rep. No. 173, Univ. of Manchester

[5] P. Deuflhard, *Newton-Techniques for Highly Nonlinear Problems — Theory and Algorithms*, erscheint bei Academic Press

[6] P. Deuflhard, *Numerik von Anfangswertmethoden für gewöhnliche Differentialgleichungen*, Vorlesungsskriptum, Technical Report TR 89–2, Konrad–Zuse–Zentrum Berlin (1989)

[7] P. Deuflhard, *Order and Stepsize Control in Extrapolation Methods*, Numer. Math. 41, 399–422 (1983)

[8] P. Deuflhard, H. J. Bauer, *A Note on Romberg Quadrature*, Preprint Nr. 169 (1982), Universität Heidelberg

[9] W. Dahmen, Angela Kunoth, *Mathematische Methoden in der geometrischen Datenverarbeitung (CAGD)*, Vorlesungsausarbeitung Freie Universität Berlin

[10] Gene H. Golub, Charles F. van Loan, *Matrix Computations*, The Johns Hopkins University Press (1989)

[11] Robert D. Skeel, *Scaling for Numerical Stability in Gaussian Elimination*, Journal of the Association for Computing Machinery, Vol. 26, No. 3, July 1979, 494–526

[12] Robert D. Skeel, *Iterative Refinement Implies Numerical Stability for Gaussian Elimination*, Mathematics of Computation, Vol. 35, No. 151, July 1980, 817–832

[13] M. Jankowski, H. Woźniakowski, *Iterative Refinement Implies Numerical Stability*, BIT 17 (1977), 303–311

[14] Konrad Knopp, *Theorie und Anwendung der unendlichen Reihen*, Springer-Verlag Berlin, Heidelberg, New York

[15] Josef Stoer, *Einführung in die Numerische Mathematik I*, Springer-Verlag Berlin, Heidelberg, New York

[16] W. Böhm, Günter Gose, Jürgen Kahmann, *Methoden der Numerischen Mathematik*, Vieweg-Verlag, 1985

[17] Hans-Peter Seidel, *Polynome, Splines und symmetrische Algorithmen im Computer Aided Geometric Design*, Habilitationsschrift Universität Tübingen, WSI–89–9

[18] Carl de Boor, *A Practical Guide to Splines*, Springer-Verlag Berlin, Heidelberg, New York

[19] Werner Romberg, *Vereinfachte Numerische Integration*, Det Kongelige Norske Videnskabers Selskabs Forhandlinger, Bind 28 (1955), Nr. 7

[20] Michael Wulkow, *Numerical Treatment of Countable Systems of Ordinary Differential Equations*, erscheint als Technical Report, Konrad–Zuse–Zentrum Berlin

[21] I. Babuška, W. C. Rheinboldt, *Error Estimates for Adaptive Finite Element Computations*, SIAM J. Numer. Anal. 15, p. 736–754 (1978)

Veröffentlichungen des Konrad-Zuse-Zentrum für Informationstechnik Berlin
Technical Reports **September 1990**

TR 86-1. H. J. Schuster. *Tätigkeitsbericht (vergriffen)*

TR 87-1. Hubert Busch; Uwe Pöhle; Wolfgang Stech. *CRAY-Handbuch. - Einführung in die Benutzung der CRAY.*
TR 87-2. Herbert Melenk; Winfried Neun. *Portable Standard LISP Implementation for CRAY X–MP Computers. Release of PSL 3.4 for COS.*
TR 87-3. Herbert Melenk; Winfried Neun. *Portable Common LISP Subset Implementation for CRAY X–MP Computers.*
TR 87-4. Herbert Melenk; Winfried Neun. *REDUCE Installation Guide for CRAY 1 / X-MP Systems Running COS Version 3.3*
TR 87-5. Herbert Melenk; Winfried Neun. *REDUCE Users Guide for the CRAY 1 / X-MP Series Running COS. Version 3.3*
TR 87-6. Rainer Buhtz; Jens Langendorf; Olaf Paetsch; Danuta Anna Buhtz. *ZUGRIFF - Eine vereinheitlichte Datenspezifikation für graphische Darstellungen und ihre graphische Aufbereitung.*
TR 87-7. J. Langendorf; O. Paetsch. *GRAZIL (Graphical ZIB Language).*

TR 88-1. Rainer Buhtz; Danuta Anna Buhtz. *TDLG 3.1 - Ein interaktives Programm zur Darstellung dreidimensionaler Modelle auf Rastergraphikgeräten.*
TR 88-2. Herbert Melenk; Winfried Neun. *REDUCE User's Guide for the CRAY 1 / CRAY X-MP Series Running UNICOS. Version 3.3.*
TR 88-3. Herbert Melenk; Winfried Neun. *REDUCE Installation Guide for CRAY 1 / CRAY X-MP Systems Running UNICOS. Version 3.3.*
TR 88-4. Danuta Anna Buhtz; Jens Langendorf; Olaf Paetsch. *GRAZIL-3D. Ein graphisches Anwendungsprogramm zur Darstellung von Kurven- und Funktionsverläufen im räumlichen Koordinatensystem.*
TR 88-5. Gerhard Maierhöfer; Georg Skorobohatyj. *Parallel-TRAPEX. Ein paralleler, adaptiver Algorithmus zur numerischen Integration ; seine Implementierung für SUPRENUM-artige Architekturen mit SUSI.*

TR 89-1. *CRAY-HANDBUCH. Einführung in die Benutzung der CRAY X-MP unter UNICOS.*
TR 89-2. Peter Deuflhard. *Numerik von Anfangswertmethoden für gewöhnliche Differential-*gleichungen.
TR 89-3. Artur Rudolf Walter. *Ein Finite-Element-Verfahren zur numerischen Lösung von Erhaltungsgleichungen.*
TR 89-4. Rainer Roitzsch. *KASKADE User's Manual.*
TR 89-5. Rainer Roitzsch. *KASKADE Programmer's Manual.*
TR 89-6. Herbert Melenk; Winfried Neun. *Implementation of Portable Standard LISP for the SPARC Processor.*
TR 89-7. Folkmar A. Bornemann. *Adaptive multilevel discretization in time and space for parabolic partial differential equations.*
TR 89-8. Gerhard Maierhöfer; Georg Skorobohatyj. *Implementierung des parallelen TRAPEX auf Transputern.*

TR 90-1. Karin Gatermann. *Gruppentheoretische Konstruktion von symmetrischen Kubaturformeln.*
TR 90-2. Gerhard Maierhöfer; Georg Skorobohatyj. *Implementierung von parallelen Versionen der Gleichungslöser EULEX und EULSIM auf Transputern.*
TR 90-3. *CRAY-Handbuch. Einführung in die Benutzung der CRAY X-MP unter UNICOS 5.1*
TR 90-4. Hans-Christian Hege. *Datenabhängigkeitsanalyse und Programmtransformationen auf CRAY-Rechnern mit dem Fortran-Präprozessor* ***fpp.***
TR 90-5. Grammel; Maierhöfer; Skorobohatij. *Trapex in POOL; Implementierung eines numerischen Algorithmus in einer parallelen objektorientierten Sprache.*
TR 90-6. Peter Deuflhard; Andreas Hohmann. *Einführung in die Numerische Mathematik.*

Veröffentlichungen des Konrad-Zuse-Zentrum für Informationstechnik Berlin
Preprints September 1990

SC 86-1. P. Deuflhard; U. Nowak. *Efficient Numerical Simulation and Identification of Large Chemical Reaction Systems.* (vergriffen) In: Ber. Bunsenges. Phys. Chem., vol. 90, 1986, 940-946
SC 86-2. H. Melenk; W. Neun. *Portable Standard LISP for CRAY X-MP Computers.*

SC 87-1. J. Anderson; W. Galway; R. Kessler; H. Melenk; W. Neun. *The Implementation and Optimization of Portable Standard LISP for the CRAY.*
SC 87-2. Randolph E. Bank; Todd F. Dupont; Harry Yserentant. *The Hierarchical Basis Multigrid Method.* (vergriffen) In: Numerische Mathematik, 52, 1988, 427-458.
SC 87-3. Peter Deuflhard. *Uniqueness Theorems for Stiff ODE Initial Value Problems.*
SC 87-4. Rainer Buhtz. *CGM-Concepts and their Realizations.*
SC 87-5. P. Deuflhard. *A Note on Extrapolation Methods for Second Order ODE Systems.*
SC 87-6. Harry Yserentant. *Preconditioning Indefinite Discretization Matrices.*

SC 88-1. Winfried Neun; Herbert Melenk. *Implementation of the LISP-Arbitrary Precision Arithmetic for a Vector Processor.*
SC 88-2. H. Melenk; H. M. Möller; W. Neun. *On Gröbner Bases Computation on a Supercomputer Using REDUCE.* (vergriffen)
SC 88-3. J. C. Alexander; B. Fiedler. *Global Decoupling of Coupled Symmetric Oscillators.*
SC 88-4. Herbert Melenk; Winfried Neun. *Parallel Polynomial Operations in the Buchberger Algorithm.*
SC 88-5. P. Deuflhard; P. Leinen; H. Yserentant. *Concepts of an Adaptive Hierarchical Finite Element Code.*
SC 88-6. P. Deuflhard; M. Wulkow. *Computational Treatment of Polyreaction Kinetics by Orthogonal Polynomials of a Discrete Variable.* (vergriffen) In: IMPACT, 1, 1989, 269-301.
SC 88-7. H. Melenk; H. M. Möller; W. Neun. *Symbolic Solution of Large Stationary Chemical Kinetics Problems.*
SC 88-8. Ronald H. W. Hoppe; Ralf Kornhuber. *Multi-Grid Solution of Two Coupled Stefan Equations Arising in Induction Heating of Large Steel Slabs.*
SC 88-9. Ralf Kornhuber; Rainer Roitzsch. *Adaptive Finite-Element-Methoden für konvektions-dominierte Randwertprobleme bei partiellen Differentialgleichungen.*
SC 88-10. S -N. Chow; B. Deng; B. Fiedler. *Homoclinic Bifurcation at Resonant Eigenvalues.*

SC 89-1. Hongyuan Zha. *A Numerical Algorithm for Computing the Restricted Singular Value Decomposition of Matrix Triplets.*
SC 89-2. Hongyuan Zha. *Restricted Singular Value Decomposition of Matrix Triplets.*
SC 89-3. Wu Huamo. *On the Possible Accuracy of TVD Schemes.*
SC 89-4. H. Michael Möller. *Multivariate Rational Interpolation: Reconstruction of Rational Functions.*
SC 89-5. Ralf Kornhuber; Rainer Roitzsch. *On Adaptive Grid Refinement in the Presence of Internal or Boundary Layers.*
SC 89-6. Wu Huamo; Yang Shuli. *MmB-A New Class of Accurate High Resolution Schemes for Conservation Laws in Two Dimensions.*
SC 89-7. U. Budde; M. Wulkow. *Computation of Molecular Weight Distributions for Free Radical Polymerization Systems.*
SC 89-8. Gerhard Maierhöfer. *Ein paralleler adaptiver Algorithmus für die numerische Integration.*
SC 89-9. Harry Yserentant. *Two Preconditioners Based on the Multi-Level Splitting of Finite Element Spaces.*
SC 89-10. Ronald H. W. Hoppe. *Numerical Solution of Multicomponent Alloy Solidification by Multi-Grid Techniques.*

SC 90-1. M. Wulkow; P. Deuflhard. *Towards an Efficient Computational Treatment of Heterogeneous Polymer Reactions.*
SC 90-2. Peter Deuflhard. *Global Inexact Newton Methods for Very Large Scale Nonlinear Problems.*
SC 90-3. Karin Gatermann. *Symbolic solution of polynomial equation systems with symmetry.*
SC 90-4. Folkmar A. Bornemann. *An Adaptive Multilevel Approach to Parabolic Equations I. General Theory & 1D-Implementation.*
SC 90-5. P. Deuflhard; R. Freund; A. Walter. *Fast Secant Methods for the Iterative Solution of Large Nonsymmetric Linear Systems.*
SC 90-6. Daoliu Wang. *On Symplectic Difference Schemes for Hamiltonian Systems.*
SC 90-7. P. Deuflhard; U. Nowak; M. Wulkow. *Recent Developments in Chemical Computing.*
SC 90-8. C. Chevalier; H. Melenk; J. Warnatz. *Automatic Generation of Reaction Mechanisms for Description of Oxidation of Higher Hydrocarbons.*