Experimentelle Mathematik

Gunther H. Peichl

Skriptum zur Vorlesung im WS1998/99

Institut für Mathematik
Karl–Franzens–Universität Graz

Vorwort

Diese Lehrveranstaltung wendet sich besonders an die Studienanfänger und Hörer des 1. Studienabschnittes. Die Studierenden sollen zu einem möglichst frühen Zeitpunkt mit grundlegenden mathematischen Konzepten und Algorithmen vertraut gemacht werden und lernen, diese mit Hilfe eines interaktiven Softwarepaketes umzusetzen. In dieser Lehrveranstaltung wird MATLAB eingesetzt, dessen Syntax besonders leicht zu erlernen ist. Der notwendige theoretische Vorgriff ist beabsichtigt und soll den Studierenden Motivation für mathematische Inhalte des 1. Studienabschnittes geben. An geeigneten Stellen soll auch die Notwendigkeit bzw. der praktische Nutzen der Theorie verdeutlich werden.

Mein besonderer Dank gilt Herrn Dr. W. Prager für das sorgfältige Korrekturlesen und für zahlreiche Verbesserungsvorschläge.

Contents

KAPITEL 1

Grundlagen aus der Analysis

1. Einführung in MATLAB

MATLAB, der Name steht für *matrix laboratory*, wurde ursprünglich geschrieben, um den Zugang zu den Matrix Softwarepaketen LINPACK und EISPACK, welche den status quo der numerischen Linearen Algebra darstellen, zu erleichtern. Im Laufe der Zeit wurden auch die graphischen Möglichkeiten verbessert und Methoden der numerischen Integration, Optimierung, und Verfahren zur Lösung gewöhnlicher Differentialgleichungen eingebaut. Eine erhebliche Erweiterung des Programmumfanges brachte Version 5. Da diese Version aber noch nicht auf allen Plattformen zur Verfügung steht und die neuen Möglichkeiten spezielle Kenntnisse erfordern, werden wir uns in der folgenden Einführung in MATLAB auf das Niveau der Version 4 beschränken.

MATLAB bietet zwei Nutzungsebenen: einfache Aufgaben löst man am besten *interaktiv*. In diesem Modus wird jede Anweisung unmittelbar nach der Eingabe vom MATLAB Kern interpretiert und ausgeführt. Für komplexere Probleme steht auch eine leistungsfähige Unterprogrammstruktur zur Verfügung. Ein Dialog mit dem MATLAB Kern, bzw. das Schreiben eines MATLAB Programmes wird durch die einfache, an die übliche mathematische Notation angelehnte Syntax erheblich erleichtert. Vorerst verwenden wir MATLAB ausschließlich interaktiv.

Beim Start von MATLAB, der je nach Betriebssystem unterschiedlich ausgelöst wird, öffnet sich ein Fenster, der sogenannte *Workspace*. Mit dem Doppelpfeil "≫" (Workspace Prompt) zeigt der Workspace an, daß er zur Verarbeitung einer Eingabe bereit ist. Das Grundelement von MATLAB ist die *Matrix*, welche keine Dimensionierung benötigt. Eine $n \times m$- Matrix ist ein 2 dimensionales Feld von Zahlen, welche in einem Raster von n Zeilen mit je m Spalten angeordnet sind. 1 dimensionale Felder nennt man *Vektoren*, genauer unterscheidet man zwischen Zeilenvektoren, das sind $1 \times n$ Matrizen, und Spaltenvektoren, das sind $n \times 1$ Matrizen. Auch *Skalare* (reelle oder komplexe Zahlen) werden als 1×1–Matrix aufgefaßt. Die allgemeine Struktur einer Anweisung ist:

(A) $$Variable = Ausdruck$$

oder einfach

Ausdruck

Im Gegensatz zur mathematischen Konvention bedeutet "= " in MATLAB nicht Gleichheit, sondern steht für Wertzuweisung: Wird eine Anweisung durch Drücken der Return-Taste an den MATLAB Kern gesandt, wird zuerst "Ausdruck" ausgewertet und dann der numerische Wert von "Ausdruck" in den Speicherplatz mit dem Namen "Variable" geschrieben. Der ursprüngliche Wert von "Variable" geht dabei verloren. Wird der Name "Variable" zum ersten Mal verwendet, wird vor der Belegung automatisch der erforderliche Speicherplatz unter dem Namen "Variable" reserviert. Daraus ergibt sich folgende Regel:

Regel 1.1. Jede rechts von"=" stehende Größe muß zum Zeitpunkt ihrer Verwendung einen alphanumerischen Wert besitzen.

"Ausdruck" in (A) steht für jeden zulässigen MATLAB Befehl. "Variable" bezeichnet einen vom Benutzer gewählten Namen einer Variablen. Dieser sollte mit Bezug auf die Bedeutung der Variablen gewählt werden. Zulässige Namen sind eine Zeichenkette bestehend aus Buchstaben, Ziffern und dem Symbol "_". Das erste Zeichen muß jedoch ein Buchstabe sein. Man beachte auch: Matlab unterscheidet Groß- und Kleinschreibung. Somit bezeichnen "Var" ,"var" , " vAr", etc. jeweils verschiedene Speicherplätze.

Beispiel 1.1. (Eingaben in MATLAB)

Anweisung	Erläuterung	
≫	y=1	MATLAB legt die Variable y an und weist ihr den Wert 1 zu.
y=		MATLAB gibt das Ergebnis der vorangehenden Anweisung aus
1		
≫	x=y	MATLAB legt die Variable x an und weist ihr den Wert 1 (= Wert von y) zu
x=		Antwort von MATLAB
1		
≫	x=x+1;	Der momentane Wert von x wird um 1 erhöht und dann der Variablen x zugewiesen. Das Semikolon am Ende der Anweisung unterdrückt die Ausgabe des Ergebnisses
≫	x	Abfrage des momentanen Wertes von x
x=		MATLAB gibt den momentanen Wert von x aus
2		

Es können auch mehrere Anweisungen in eine Zeile geschrieben werden. Jede Anweisung ist dann entweder durch einen Beistrich (Ausgabe des Ergebnisses) oder durch ein Semikolon (Unterdrücken der Ergebnisausgabe) abzuschließen. Paßt eine Anweisung nicht in eine Bildschirmzeile, können mit "..." Fortsetzungszeilen geöffnet werden:

BEISPIEL 1.2. (Fortsetzung des Beispiels)

```
≫ x=x+3,y=x+y;z=y+1
x=                          MATLAB gibt wegen des Beistrichs das Ergebnis der
                            Anweisung x=x+3 aus
  5
z=                          MATLAB gibt das Ergebnis der letzten Anweisung
                            aus, die Ausgabe des Wertes von y wurde unterdrückt
  7
≫ x= ...                    Eine Fortsetzungszeile wird geöffnet
  x-y;
≫ u=5e-10;                  u wird der Wert 5 · 10^-10 zugewiesen
```

Matrizen werden eingegeben, indem man die einzelnen Elemente der Matrix zwischen "[" und "]" schreibt. Die Matrixelemente werden durch Leerzeichen bzw. durch einen Beistrich voneinander getrennt, die einzelnen Zeilen einer Matrix durch ein Semikolon:

BEISPIEL 1.3. Eingabe von Matrizen:

```
≫ x = [ 1 2 3]          x ist eine 1 × 3–Matrix
x=
    1 2 3
≫ y= [ 3 ; 4 ;5]        y ist eine 3 × 1–Matrix
y=
    3
    4
    5
≫ a=[ 1 2 3;3,4,5]
a=
    1   2   3
    3   4   5
≫ z=y'                  ' vertauscht Zeilen und Spalten
z=
    3   4   5
≫ b= [x;z]
b=
    1   2   3
    3   4   5
```

```
≫  c=b'
c=
     1   3
     2   4
     3   5
```

Es wurde auch der *Transponierungsoperator* " ' " eingeführt. Ist M eine $n \times m$–Matrix, dann ist M' eine $m \times n$-Matrix, welche aus M durch Vertauschen von Zeilen und Spalten hervorgeht. Ist M komplex, werden zusätzlich die Matrixelemente konjugiert komplex genommen.

Auf die einzelnen Elemente einer Matrix wird über ihre Position im Feld zugegriffen: wurde etwa eine Matrix M erzeugt, bezeichnet M(i,j) das Feldelement, welches in der i-ten Zeile in der j-ten Spalte steht. Für 1-dimensionale Matrizen (*Vektoren*) V sind die Zugriffe V(1,i) bzw. V(i,1) gleichwertig mit V(i). Hat V jedoch die Dimension $1 \times N$, ergibt der Zugriff V(i,1) für $i > 1$ die Fehlermeldung "Index exceeds matrix dimensions", da V ja nur 1 Zeile besitzt. Es können auch Vektoren für die Indizierung verwendet werden. Dies ist eine effiziente Methode, um aus Matrizen bestimmte Teilmatrizen herauszugreifen. Ist beispielsweise A eine 10×10–Matrix, dann wird durch

```
≫ B = A(2:3:10,3:6);
```

eine 3×4–Matrix B angelegt (falls die Variable B im Workspace noch nicht existiert hat). In den Zeilen der Matrix B stehen die Matrixelemente von A aus den Zeilen 2,5,und 8 und der Spalten 3,4,5 und 6. Nimmt man nur den Doppelpunkt als Index, werden sämtliche Elemente der betroffenen Zeilen oder Spalten genommen: mit A(:,3:6) kann man auf die Spalten 3-6 der Matrix A zugreifen, A(5,:) ist die gesamte 5. Zeile von A.

1.1. Feldoperationen für Matrizen. Es können nicht nur einzelne Matrixelemente in Rechnungen verwendet werden (wie etwa in FORTRAN), sondern MATLAB ermöglicht auch die Verknüpfung ganzer Matrizen:

+	Addition
-	Subtraktion
.*	Multiplikation
./	Division
.^	Potenz

(Auf manchen Tastaturen muß nach "^" ein Leerzeichen eingegeben werden, andernfalls erhält man z.B. $2\hat{3}$ anstelle von 2^3.) Man achte auf den vorgesetzten Punkt bei der Multiplikation, Division und Potenz, da diese Operatoren auch ohne Punkt, jedoch mit vollkommen unterschiedlicher Wirkung, definiert sind. Diese Feldoperationen verknüpfen jeweils Matrixelemente mit gleichen Indizes. Beispielsweise ergibt die Feldmultiplikation (nicht zu verwechseln mit der Matrizenmultiplikation in der

Linearen Algebra) der 2×2–Matrizen A, B

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad B = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$$

die Matrix

$$A.*B = \begin{bmatrix} a\alpha & b\beta \\ c\gamma & d\delta \end{bmatrix}.$$

Die Matrixelemente $a, \ldots, d$ bzw. $\alpha, \ldots, \delta$ sind entweder Zahlen oder Variable, welchen bereits ein numerischer Wert zugewiesen wurde. Naturgemäß müssen daher die verknüpften Matrizen dasselbe Format haben oder einer der Operanden A 0der B ist ein Skalar.

Beispiel 1.4. (Feldoperationen)

```
≫ x=[1 2 3]; y=[3;4;5]
≫ y-x                          x und y haben unterschiedliches Format!
??? Error using ==> -
Matrix dimensions must agree   Fehlermeldung von MATLAB
≫ y=y'                         " ' " vertauscht Zeilen und Spalten
y=
    3 4 5                      y wurde als Zeile formatiert
≫ y-x                          In dieser Anweisung erfolgt keine
                               Wertzuweisung
ans =                          MATLAB legt die Hilfsvariable ans für die
                               Wertzuweisung an
    2 2 2
≫ y.*x
ans =
    3 8 15
≫ y./x
ans =
    3.000 2.0000 1.6667
≫ y.^x
ans =
    3 16 125
≫ 2*x                          gleichwertig mit 2.*x oder x*2 oder x.*2
ans =
    2 4 6
≫ x/2                          gleichwertig x./2
ans =
    .5000 1.0000 1.5000
```

```
    ≫  2./x                    2/x ergibt eine Fehlermeldung
ans =
        2.0000 1.0000 0.6667
    ≫  x.^2
ans =
        1 4 9
    ≫  2.^x
ans =
        2 4 8
```

Die Beispiele zeigen, daß MATLAB bei der Eingabe eines Ausdruckes ohne Wertzuweisung auf eine Variable standardmäßig eine passende Hilfsvariable "ans" erzeugt, auf welche die Wertzuweisung erfolgt. Oft wird ein Zahlenfeld mit der Struktur einer arithmetischen Progression benötigt. Ausgehend von einem Startwert "start" werden die einzelnen Feldelemente durch sukkzessive Addition eines festen Inkrementes "inc" erzeugt, bis ein vorgegebener Endwert "end" erreicht (aber nicht überschritten) wird. Ein solches Zahlenfeld wird durch die Anweisung

start:inc:end

berechnet. Das Ergebnis ist immer ein Zeilenvektor passender Länge. Man kann auch

start:end

schreiben, falls $inc = 1$ gilt. Ist $inc > 0$ und $start > end$, gibt es keine reellen Zahlen, die diesen Vorgaben entsprechen: das erzeugte Feld ist leer. Das leere Feld wird in MATLAB mit "[ ]" bezeichnet.

Beispiel 1.5. (Erzeugen arithmetischer Progressionen)

```
≫     x=1:2:7
x =
       1 3 5 7
≫     y=1:2:6
y =
         1 3 5
≫    z= 9:-3:2   Das Inkrement kann auch negativ sein
z=
         9 6 3
≫      u= 1:4
u=
       1 2 3 4
≫    v = 3:2:1
v=
            [ ]
```

Selbstverständlich sind in MATLAB die gängigen Funktionen implementiert, z.B.

mathematische Notation	MATLAB
$\|x\|$	abs(x)
$\sin(x)$	sin(x)
$\cos(x)$	cos(x)
$\tan(x)$	tan(x)
$\ln(x)$	log(x)
e^x	exp(x)
$\sqrt{x}$	sqrt(x)
a^x	a.^x

TABELLE 1.1. Funktionen in MATLAB

MATLAB stellt auch reichhaltige online Information zur Verfügung. Mit dem Befehl "help" erhält man eine Liste von Kategorien, für welche es online Hilfe gibt. Kennt man die MATLAB Anweisung, kann man sich mit "help Anweisung" über Wirkung , Gebrauch und mögliche Optionen informieren.

1.2. 2d- Graphik. Eine sehr wesentliche Aufgabe ist die graphische Darstellung einer Funktion auf einem vorgegebenen Intervall $[a, b]$. MATLAB bietet dazu 2 Möglichkeiten an:

Variante 1:

1. Erzeugen eines Feldes von Stützstellen, an denen die Funktion ausgewertet werden soll. Im einfachsten Fall wählt man äquidistante Stützstellen, z.B.
   ≫ x= a:(b-a)/100:b
2. Auswertung der Funktion in x und Abspeichern der Funktionswerte in y
3. Erzeugen des Funktionsgraphen durch
   ≫plot(x,y)

Diese Methode ist angebracht, wenn die Funktion bereits an den Stützstellen ausgewertet wurde oder die Funktionswerte nur für einzelne Werte von x, etwa bei Meßdaten, zur Verfügung stehen.

Variante 2:

≫ fplot('f(x)',[a,b])

In dieser Anweisung steht "f(x)" für die MATLAB Anweisung zur Auswertung von f(x). In dieser Anweisung darf nur eine Variable vorkommen. Wegen der größeren Flexibilität wird meist Variante 1 verwendet.

Über "help plot" erfährt man, daß der Linientyp für die Erstellung der Graphik durch " plot(x,y,'typ')" festgelegt wird:

- \- durchgezogen (Standard)
- : punktiert
- -- strichliert
- -. strichpunktiert

BEISPIEL 1.6. Simultane Darstellung der Funktionen $f(x) = \frac{1}{2}\sin(3\pi x)$ und $g(x) = \frac{1}{2}\cos(3\pi x)$ auf dem Intervall $[0, \pi]$ in einer Graphik. Zur Unterscheidung der beiden Funktionen soll der Graph von g strichliert dargestellt werden. Der Einfachheit halber wird im Folgenden der Workspace Prompt $\gg$ nicht mehr angeführt.

x=0:pi/100:pi;	Erzeugen des Feldes der Stützstellen, Zugriff auf π mit pi
f=.5*sin(3*pi*x);	Auswertung von f
g=.5*cos(3*pi*x);	Auswertung von g
plot(x,f)	
hold on	Einfrieren des Graphikschirmes: alle nachfolgenden Plotanweisungen betreffen die momentane Graphik
plot(x,g,'--')	Overlay des Graphen von g, Linientyp strichliert
hold off	Freigabe des Graphikschirmes: eine weitere Plotanweisung löscht die momentane Graphik
title('Überschrift')	Man achte auf die Apostrophe
xlabel('x Achse')	
ylabel('y Achse')	
clg	Löscht den Graphikschirm
plot(x,f,x,g,'--')	Erzeugt dieselbe Graphik (ohne Beschriftung)

Abb. 1.1 zeigt die fertige Graphik. Da MATLAB standardmäßig die Achsen so skaliert, daß die Graphik das Graphikfenster möglichst gut ausfüllt, kommt es oft zu einer beträchtlichen Verzerrung der Proportionen. Diese kann durch die Anweisung "axis('equal')" korrigiert werden. "axis('normal')" schaltet wieder in den Standardmodus zurück. Abb. 1.2 veranschaulicht die Wirkung von "axis('equal')".

1.3. 3d-Graphik. Die graphische Darstellung von Funktionen in 2 Veränderlichen, bzw. von Flächen im 3–dimensionalen Raum über einem Rechteck $[a, b] \times [c, d]$ erfolgt analog. Anstelle eines Vektors von Stützstellen ist nun ein Gitter von Stützstellen (x_i, y_j) zu erzeugen. Am einfachsten läßt sich ein in jeweils einer Achsenrichtung äquidistantes Gitter erzeugen.

BEISPIEL 1.7. Darstellung von $f(x, y) = \frac{\sin(x^2+y^2)}{x^2+y^2}$, $(x, y) \neq (0, 0)$, $f(0, 0) = 0$ auf $[-8, 8] \times [-8, 8]$. Die folgende Programmsequenz ermöglicht die Auswertung von f mit einem einzigen Befehl in *allen* Gitterpunkten (x_i, y_j).

x = -8:.5:8;	
[X,Y]= meshgrid(x,x);	gleiche Maschenweite in beiden Koordinatenrichtungen, äquivalent: [X,Y] = meshgrid(x);
r = sqrt(X.^2+Y.^2) + eps;	Addition von eps verhindert Division durch Null
z=sin(r)./r;	Auswertung von f auf dem gesamten Gitter
mesh(x,x,z)	Erzeugen der 3d–Graphik, zur Skalierung beider Koordinatenachsen wird der Vektor x verwendet

eps, das sogenannte Maschinenepsilon, ist die kleinste positive Zahl, welche auf dem Computer die Ungleichung

$$1 < 1 + \text{eps}$$

erfüllt. Auf einem PC gilt meist eps = $2.22 \cdot 10^{-16}$. Diese Maschinenkonstante ist in MATLAB implementiert und sollte nicht neu definiert werden.

Die Anweisung "[X,Y] = meshgrid(x,y)" erzeugt die Matrizen X und Y, deren Spaltenanzahl der Länge von x und deren Zeilenanzahl der Länge von y entspricht. Die Zeilen von X sind Kopien des Vektors x, die Spalten von Y sind Kopien von y. Der Wert von $r(i,j) = x_i^2 + y_j^2$ entspricht daher dem (j,i)–ten Element von X.^2 + Y.^2. Für die graphische Darstellung von z wird dann noch die Reihenfolge der Zeilen umgekehrt, sodaß der Punkt (x_1, y_1) in die(perspektivisch gesehen) linke, vordere Ecke des Graphikfensters zu liegen kommt.

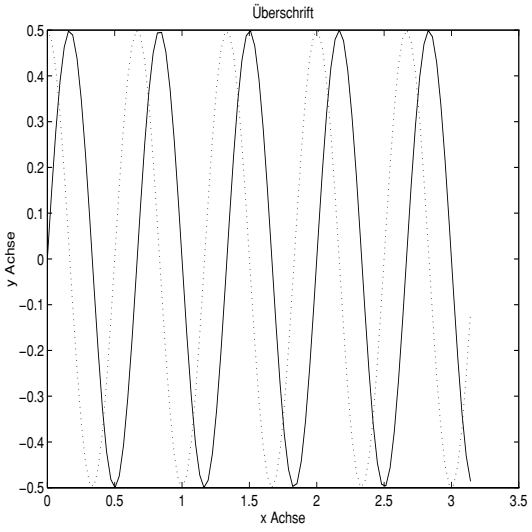


ABB. 1.1. Automatische Skalierung

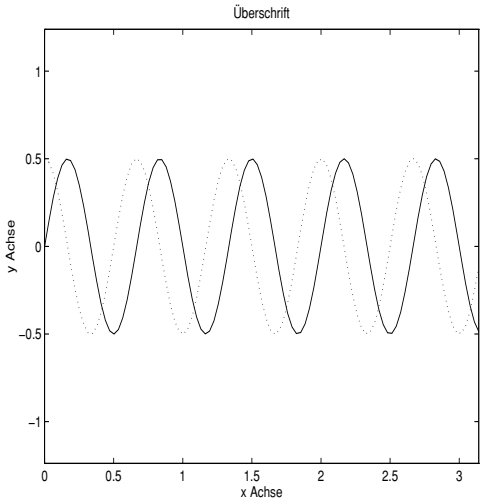


ABB. 1.2. Ohne automatische Skalierung

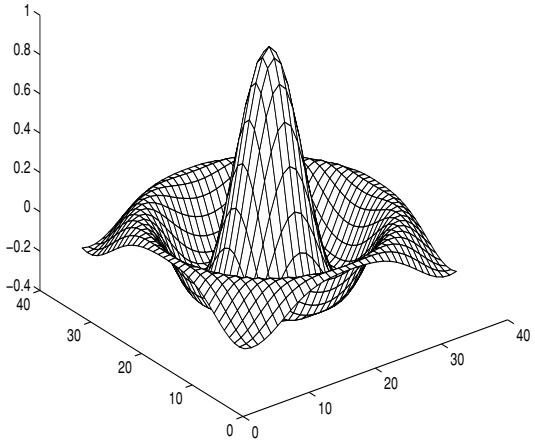


Abb. 1.3. Sombrero

1.4. Der Befehlszeilenspeicher. Bei längeren MATLAB Sitzungen wird man häufig Anweisungen eingeben, welche sich voneinander nur geringfügig unterscheiden, z.B. bei der Korrektur einer fehlerhaften Eingabe. MATLAB unterhält einen Befehlszeilenspeicher, der es ermöglicht, mit Hilfe der Pfeiltasten ↑ und ↓ im Laufe der Sitzung eingegebene Zeilen zu suchen. Gibt man eine charakteristische Zeichenkette des Zeilenbeginns an, werden nur jene Befehlszeilen angezeigt, welche mit exakt dieser Zeichenkette beginnen. Ist der gesuchte Befehl gefunden, kann die Eingabe mit den Tasten des mittleren Tastaturblockes editiert werden. Es stehen folgende Funktionen zur Verfügung:

←	bewegt Cursor um 1 Zeichen nach links
→	bewegt Cursor um 1 Zeichen nach rechts
Home	bewegt Cursor an den Zeilenanfang
End	bewegt Cursor an das Zeilenende
Esc	löscht die Befehlszeile
Backspace	löscht das Zeichen links vom Cursor
Del	löscht das Zeichen unter dem Cursor

2. Der Konvergenzbegriff

Der zentrale Begriff in der Analysis ist die Konvergenz einer Folge reeller oder komplexer Zahlen. Im Folgenden steht $\mathbb{K}$ stets für den Körper der reellen oder komplexen Zahlen. Eine **Folge** $(x_n)_{n=1}^{\infty} \subset \mathbb{K}$ ist eine Abbildung $x\colon \mathbb{N} \to \mathbb{K}$. Anstelle von $x(n)$ schreibt man jedoch allgemein x_n.

DEFINITION 2.1. *Eine Folge* $(x_n)_{n=1}^{\infty} \subset \mathbb{K}$ *heißt* **konvergent gegen** $\mathbf{x_0} \in \mathbb{K}$ *genau dann, wenn es zu jedem* $\varepsilon > 0$ *einen Index* $N(\varepsilon)$ *gibt, sodaß*

$$|x_n - x_0| < \varepsilon$$

für alle $n \geq N(\varepsilon)$ *zutrifft. Dies wird durch die Schreibweise*

$$x_0 = \lim_{n\to\infty} x_n$$

ausgedrückt. Man nennt x_0 **Grenzwert** *der Folge. Eine Folge heißt* **konvergent**, *wenn es ein* $x_0 \in \mathbb{K}$ *gibt, sodaß* $x_0 = \lim_{n\to\infty} x_n$ *gilt. Eine Folge heißt* **divergent**, *wenn sie nicht konvergent ist.*

Anschaulich wird in dieser Definition gefordert, daß in jedem Intervall $(x_0-\varepsilon, x_0+\varepsilon)$, *egal wie klein* $\varepsilon > 0$ *gewählt wurde*, alle Folgenglieder mit höchstens endlich vielen Ausnahmen liegen. Eine offensichtliche Schwierigkeit bei der Untersuchung der Konvergenz einer Folge ist, daß in der Definition 2.1 die Kenntnis des Grenzwertes x bereits vorausgestzt wird. Man hat daher Konvergenzkriterien entwickelt, welche nur "innere" Eigenschaften der Folge benützen. Das Allgemeinste dieser Kriterien ist das Cauchykriterium, auf das hier nicht eingegangen werden kann. Für monotone Folgen reeller Zahlen ist das Monotoniekriterium anwendbar:

DEFINITION 2.2.
1. *Eine Folge* $(x_n)_{n=1}^{\infty} \subset \mathbb{K}$ *heißt* **beschränkt** *genau dann, wenn es ein* $M > 0$ *gibt mit*

$$|x_n| \leq M \qquad \text{für alle } n \in \mathbb{N}.$$

2. *Eine Folge* reeller *Zahlen heißt* **monoton wachsend** *(***monoton fallend***) genau dann, wenn*

$$x_n \leq x_{n+1} \qquad (x_n \geq x_{n+1})$$

für alle $n \in \mathbb{N}$ *zutrifft.*
3. *Eine Folge reeller Zahlen heißt* **monoton** *genau dann, wenn sie monoton wachsend oder monoton fallend ist.*

THEOREM 2.1 (Monotoniekriterium). *Eine monotone Folge ist genau dann konvergent, wenn sie beschränkt ist.*

Vermutungen, ob eine gegebene Folge konvergiert oder divergiert, aber auch Ideen für Beweisansätze lassen sich oft aus einer graphischen Darstellung eines Anfangsstükkes der Folge ablesen. Will man eine bestimmte Anzahl von Folgengliedern berechnen, wird eine Rechenoperation oder eine Gruppe von Rechenoperationen immer wiederholt. Bei jedem Durchgang ist lediglich n durch $n+1$ zu ersetzen. Dies kann durch eine *for*-Schleife mit folgender Syntax automatisiert werden:

for i = start : ink : ende
⋮
Anweisungen

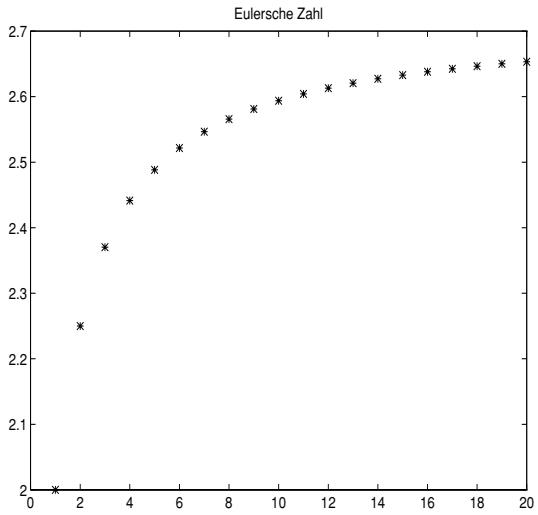


ABB. 2.1. Eulersche Zahl

⋮
end

Der Standardwert von "ink" ist 1 und muß nicht angeführt werden: "for i = start : 1 : ende" ist gleichwertig mit "i = start : ende". Ist $ink > 0$ wird diese Sequenz folgendermaßen abgearbeitet: Im Falle $ende < start$ überspringt MATLAB den for-Block und führt die erste Anweisung nach "end" aus. Im Regelfall ist $start < ende$ und MATLAB führt die zwischen "for" und "end" stehenden Anweisungen mit $i = start$ aus. Beim Erreichen von "end" wird i durch $i + ink$ ersetzt und die Bedingung $i \leq ende$ abgefragt. Ist sie erfüllt, springt MATLAB zur ersten Anweisung des for–Blockes zurück, andernfalls wird der for–Block verlassen.

BEISPIEL 2.1. Es soll die Konvergenz der Folge $x_n = (1 + \frac{1}{n})^n$, $n \in \mathbb{N}$, untersucht werden. Wir veranschaulichen zuerst das qualitative Verhalten dieser Folge und berechnen die ersten 20 Folgenglieder:

```
N=20;                    Anzahl der zu berechnenden Folgenglieder
x=zeros(N,1);            Initialisiere x als Spaltenvektor der Länge N, jedes Element
                         hat den Wert 0
for i=1:N                i wird bei jedem Durchgang um 1 vergrößert
   x(i)=(1+1/i)^i;       Einrücken erleichtert die Lesbarkeit der for–Schleife
end
plot(x,'*')
title('Eulersche Zahl')  Erzeugt Figur 2.1
```

Es wäre nicht notwendig, den Vektor x zu initialisieren. Die Schleife wird dadurch jedoch schneller abgearbeitet, da andernfalls bei jedem Schleifendurchgang der Vektor x neu dimensioniert werden müßte. Man achte auch auf die geänderte Form der "plot"–Anweisung: "plot(x)" erzeugt eine Graphik, in der die Werte von x gegen die Indizes der Elemente von x aufgetragen sind. "plot(x,'∗')" bewirkt, daß an die Stellen $(i, x(i))$ lediglich das Symbol $*$ gesetzt wird, die einzelnen Datenpunkte jedoch nicht durch einen kontinuierlichen Linienzug verbunden werden. Als Marker stehen zur Verfügung

.	Punkt	o	Kreis
x	Kreuz	+	Plus
$*$	Stern		

In diesem Beispiel kann das Anfangsstück der Folge wesentlich effizienter mit Hilfe der Feldoperationen erzeugt werden:

```
N=20;i=1:N;x=(1+1./i).^i;
```

Der Einsatz einer for-Schleife sollte nach Möglichkeit vermieden und durch Gebrauch geeigneter Feldoperationen ersetzt werden.

Abb. 2.1 legt die Hypothese nahe, daß die Folge (x_n) monoton wächst und beschränkt ist. Falls die Hypothese zutrifft, konvergiert die Folge nach dem Monotoniekriterium. Berechnet man weitere Folgenglieder, kann man für den Grenzwert $x_0 \approx 2.7$ vermuten. Wie in der Analysis bewiesen wird, konvergiert diese Folge tatsächlich. Der Grenzwert wird **Eulersche Zahl e** $= 2.71828182845904\cdots$ genannt. Dieses Beispiel veranschaulicht auch die Schwäche der Definition 2.1. Natürlich kann dieses Vorgehen nicht einen exakten Konvergenzbeweis ersetzen. Die numerische Untersuchung legt aber folgendes analytische Vorgehen nahe:

- Nachweis, daß (x_n) monoton wächst,
- Nachweis der Beschränktheit
- Folgerung der Konvergenz.

Beispiel 2.2. Als Beispiel einer *rekursiv definierten* Folge $(b_n)_{n=1}^{\infty}$ betrachten wir

$$b_1 = 0,$$
$$b_n = \frac{b_{n-1}}{2} + \frac{(-1)^{n-1}}{2^{n-1}}, \quad n \geq 2.$$

Berechnen wir wieder die ersten 20 Glieder dieser Folge. Um den Einfluß des wechselnden Vorzeichens in der Rekursionsvorschrift besser verfolgen zu können, markieren wir die Folgenglieder mit geradem Index mit "+", jene mit ungeradem Index mit "o". Da zur Berechnung des n-ten Folgengliedes nach obiger Formel die Kenntnis des (n-1)-ten Folgengliedes erforderlich ist, läßt sich eine for–Schleife nicht vermeiden.

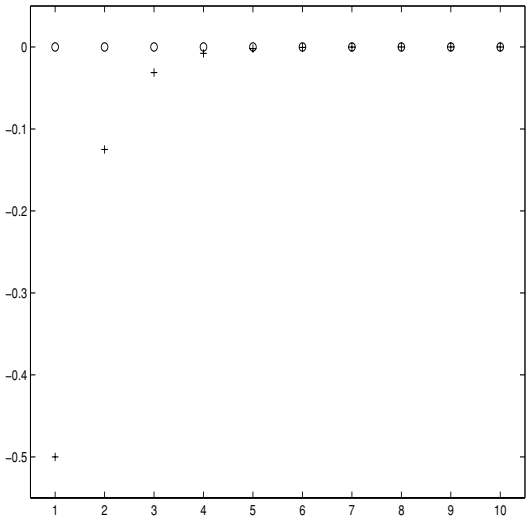


ABB. 2.2. $b_n = \frac{b_{n-1}}{2} + \frac{(-1)^{n-1}}{2^{n-1}}$

```
b=zeros(20,1);
s=1;                    Hilfsvariable
for i=2:20
s=-s/2;                 bewirkt den Vorzeichenwechsel und den Aufbau von 1/2^(n-1)
b(i) = b(i-1)/2+s;
end
plot(b(1:2:20),'o')     Darstellung der ungeradzahlig indizierten Folgenglieder
hold on                 Einfrieren des Graphikschirmes
plot(b(2:2:20),'+')
```

Abb. 2.2 zeigt ein überraschendes Verhalten dieser Folge: Offenbar gilt

$$b_{2n+1} = 0,$$

für alle $n \in \mathbb{N} \cup \{0\}$. Kombiniert mit der Rekursionsvorschrift hätte dies jedoch

$$b_{2n} = -\frac{1}{2^{2n-1}}, \qquad \text{für alle } n \in \mathbb{N}$$

zur Folge. Ein Blick auf die numerischen Werte von b_n bestätigt diesen Verdacht. Es ist nun nicht mehr schwer, mit Hilfe vollständiger Induktion und der Rekursionsvorschrift die Vermutung

$$b_n = \begin{cases} 0 & \text{für } n \text{ ungerade} \\ -\frac{1}{2^{n-1}} & \text{für } n \text{ gerade} \end{cases}$$

auch zu beweisen.

2.1. Konvergenzordnung. Als Ergebnis vieler numerischer Verfahren wird eine Folge reeller Zahlen berechnet. Bei einem sinnvollen Verfahren sollte diese Folge natürlich zumindest konvergieren. In der Praxis ist allerdings auch wesentlich, wie rasch eine gute Approximation des Grenzwertes erzielt wird. Zur Beschreibung der Konvergenzgeschwindigkeit führen wir den Begriff der Konvergenzordnung (oder Konvergenzrate) ein.

DEFINITION 2.3 (Konvergenzordnung). *Es sei $(x_n) \subset \mathbb{K}$ eine konvergente Folge mit Grenzwert ξ und $C > 0$.*

1. *Die Folge (x_n) konvergiert gegen ξ mit der* **Konvergenzordnung $\mathbf{p} > 0$** *genau dann, wenn*

$$\lim_{n\to\infty} \frac{|x_{n+1} - \xi|}{|x_n - \xi|^p} = C. \tag{2.1}$$

2. *Die Fälle $p = 1$, $p = 2$ bzw. $p = 3$ bezeichnet man als* **lineare, quadratische** *bzw.* **kubische Konvergenz**

Bei linearer Konvergenz muß notwendigerweise $C \leq 1$ sein, da andernfalls die Differenzen $x_{n+1} - x_n$ keine Nullfolge bilden können. Ist der Grenzwert ξ nicht bekannt, kann man eine Hypothese über die Konvergenzordnung durch Untersuchung des Quotienten

$$\frac{|x_{n+1} - x_n|}{|x_n - x_{n-1}|^p}$$

aufstellen.

BEISPIEL 2.3. Wir betrachten die rekursiv definierte Folgen

$$\begin{aligned} x_0 &= 2 = y_0 \\ x_k &= x_{k-1} + 10^{-k}, \quad k \geq 1 \\ y_k &= y_{k-1} + 10^{-2^{k-1}}, \quad k \geq 1, \end{aligned}$$

für welche offensichtlich

$$\frac{x_{k+1} - x_k}{x_k - x_{k-1}} = 0.1 \quad \text{bzw.} \quad \frac{y_{k+1} - y_k}{(y_k - y_{k-1})^2} = \frac{10^{-2^k}}{(10^{-2^{k-1}})^2} = 1$$

gilt. Die Folge (x_n) konvergiert daher linear (gegen $\xi = 2\frac{1}{9}$), die Folge (y_n) konvergiert quadratisch (gegen $\xi = 2 + \sum_{k=0}^{\infty} 10^{-2^k}$). Tabelle 2.1 veranschaulicht den Unterschied zwischen linearer und quadratischer Konvergenz. Als Faustregel gilt: Konvergiert eine Folge linear, dann benötigt man im Mittel jeweils dieselbe Anzahl von Folgengliedern, um 1 Dezimalstelle des Grenzwertes zu berechnen. Bei quadratischer Konvergenz verdoppeln sich die mit dem Grenzwert übereinstimmenden Dezimalstellen im Mittel nach jeweils derselben Anzahl von Folgengliedern. Im Beispiel

$x_0 = 2$	$y_0 = 2$
$x_1 = 2.1$	$y_1 = 2.1$
$x_2 = 2.11$	$y_2 = 2.11$
$x_3 = 2.111$	$y_3 = 2.1101$
$x_4 = 2.1111$	$y_4 = 2.11010001$
$x_5 = 2.11111$	$y_5 = 2.1101000100000001$
$x_6 = 2.111111$	$y_6 = 2.11010001000000010000000000000001$

TABELLE 2.1. Lineare und quadratische Konvergenz

stimmt jedes Folgenglied x_k in k Stellen, y_k allerdings in 2^{k-1} Stellen mit dem Grenzwert überein.

2.2. Stetige Funktionen. Eng verbunden mit der Konvergenz von Folgen ist der Begriff des Grenzwertes einer Funktion an einer Stelle x_0. Der Einfachheit halber sei $I \subset \mathbb{R}$ ein Intervall und $x_0 \in I$ oder ein Randpunkt von I. Eine Abbildung f sei auf I oder $I \setminus \{x_0\}$ definiert. Ist die Funktion in x_0 nicht definiert, ist man oft daran interessiert, den Definitionsbereich der Funktion zu erweitern, indem man an der Stelle x_0 einen sinnvollen Funktionswert festsetzt. In diesem Fall will man feststellen, ob die Wertzuweisung in x_0 mit den benachbarten Funktionswerten "harmoniert".

DEFINITION 2.4. *Es sei I ein Intervall mit den Randpunkten $a < b$ und $x_0 \in [a, b]$. $F \in \mathbb{K}$ heißt* **Grenzwert von $\mathbf{f}\colon \mathbf{I} \setminus \{\mathbf{x_0}\} \to \mathbb{K}$ an der Stelle $\mathbf{x_0}$** *genau dann, wenn für* jede *Folge $(x_n) \subset I \setminus \{x_0\}$ mit $\lim_{n\to\infty} x_n = x_0$*

$$\lim_{n\to\infty} f(x_n) = F$$

zutrifft. Man schreibt $\lim_{x\to x_0} f(x) = F$.

Wenn der Grenzwert von f in x_0 existiert, stellt er den "natürlichen" Wert für die Fortsetzung von f nach x_0, bzw. für die Definition von f in x_0 dar. Naturgemäß kann man die Bedingung in Definition 2.4 am Rechner nicht simulieren, da es unmöglich ist, *alle* nach x_0 konvergenten Folgen zu testen. Allerdings gibt der Funktionsgraph meist genügend Aufschluß für einen analytischen Ansatz.

BEISPIEL 2.4. Als Beispiel untersuchen wir die Grenzwerte der Funktionen

$$f(x) = \begin{cases} \frac{\sin x}{x}, & x \neq 0 \\ 1, & x = 0 \end{cases} \qquad g(x) = \begin{cases} \frac{e^x - 1}{|x|}, & |x| \neq 0 \\ 0, & x = 0 \end{cases}$$

Abbildung 2.3 legt nahe, daß $\lim_{x\to 0} f(x) = 1$ existiert, während Abb. 2.4 vermuten läßt, daß die Funktion g in $x = 0$ keinen Grenzwert besitzt. Hätten wir also f in $x = 0$ nicht definiert, wäre $f(0) = 1$ die "natürliche" Wahl für die Fortsetzung von

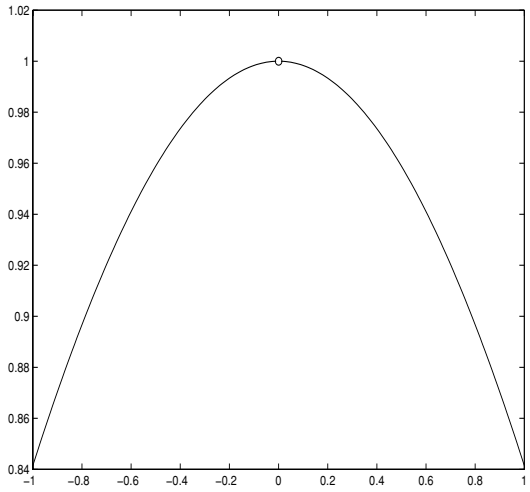


ABB. 2.3. $f(x) = \frac{\sin x}{x}$

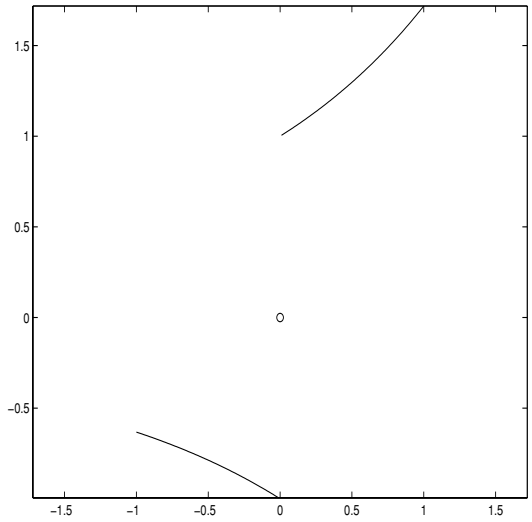


ABB. 2.4. $g(x) = \frac{e^x - 1}{|x|}$

f gewesen. Für g hingegen gibt es offensichtlich keine "natürliche" Fortsetzung nach $x = 0$.

Bestehen wir nun darauf, daß die Abbildung an der Stelle x_0 tatsächlich definiert sein soll und wollen wir klären, was wir in diesem Zusammenhang mit "natürlich" meinen, gelangen wir zum Begriff der Stetigkeit von f in x_0:

DEFINITION 2.5. *Es sei I ein Intervall und $x_0 \in I$. Eine Funktion $f\colon I \to \mathbb{R}$ heißt* **stetig in $\mathbf{x_0}$** *genau dann, wenn f an der Stelle x_0 einen Grenzwert besitzt und*

$$f(x_0) = \lim_{x \to x_0} f(x)$$

gilt. Eine Funktion f heißt stetig, wenn sie in jedem Punkt ihres Definitionsbereiches stetig ist.

Stetigkeit einer Funktion ist im Folgenden stets eine Minimalvoraussetzung, denn diese Eigenschaft stellt sicher, daß sich bei der numerischen Auswertung einer Funktion Rundungsfehler nicht zu stark auswirken können. Stetige Funktionen haben auch nützliche Eigenschaften, von denen wir nur folgende notieren:

THEOREM 2.2 (Zwischenwertsatz). *Eine stetige Funktion bildet ein Intervall stets wieder auf ein Intervall ab.*

Gilt somit für ein $\lambda \in \mathbb{R}$ die Ungleichung $f(a) < \lambda < f(b)$ und bezeichnet $I(a, b)$ das Intervall mit den Endpunkten a und b (es muß nicht notwendig $a < b$ gelten), dann ist nach dem Zwischenwertsatz $f(I(a, b))$ selbst wieder ein Intervall. Somit muß λ im Bild von $I(a, b)$ liegen (was a priori nicht klar ist). Jeder Wert zwischen zwei Funktionswerten einer stetigen Funktion kommt also selbst wieder als Funktionswert vor. Dies erklärt die Bezeichnung Zwischenwertsatz für Theorem 2.2.

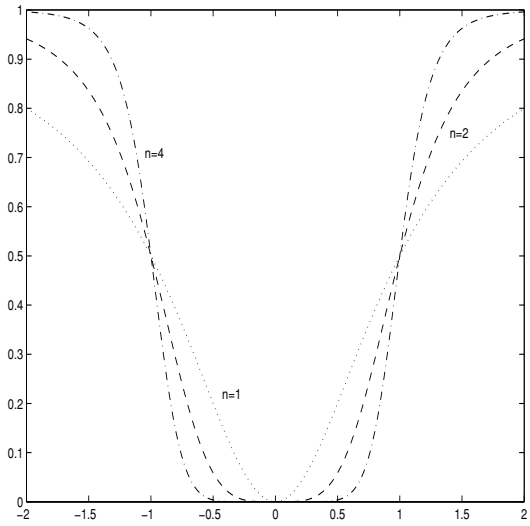


ABB. 2.5. $f_n(x) = \frac{x^{2n}}{1+x^{2n}}$

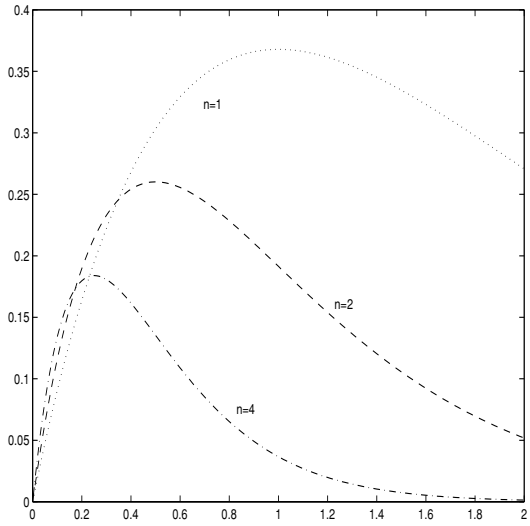


ABB. 2.6. $g(x) = \sqrt{n}xe^{-nx}$

THEOREM 2.3 (Satz von Weierstraß). *Jede stetige Funktion nimmt auf einem abgeschlossenen Intervall* $[a, b]$, $-\infty < a \leq b < \infty$ *Maximum und Minimum an.*

Der Satz von Weierstraß und seine Verallgemeinerungen sichern die Existenz einer Lösung für zahlreiche Optimierungsprobleme.

2.3. Funktionenfolgen. Es sei (f_n), $f_n \colon I \to \mathbb{R}$, $n \in \mathbb{N}$, eine Folge von Funktionen. Es ist naheliegend, die Konvergenz einer Funktionenfolge dadurch zu charakterisieren, daß für jedes $x \in I$ die Folge der Funktionswerte $(f_n(x))$ konvergiert. Dies ist ein sehr schwacher Konvergenzbegriff, da er den Funktionsaspekt nicht entsprechend berücksichtigt. Es ist auch nicht schwer, Beispiele von Funktionenfolgen mit qualitativ recht unterschiedlichem Konvergenzverhalten zu finden.

BEISPIEL 2.5. Die Funktionen $f_n \colon \mathbb{R} \to \mathbb{R}$ und $g_n \colon [0, \infty) \to \mathbb{R}$ seien definiert durch

$$f_n(x) = \frac{x^{2n}}{1 + x^{2n}}$$
$$g_n(x) = \sqrt{n}xe^{-nx}.$$

Abb. 2.5 zeigt das qualitative Verhalten der Funktionenfolge (f_n): Läßt man n immer größer werden nähern sich die Funktionswerte $f_n(x)$ für $|x| > 1$ dem Wert 1, für $|x| < 1$ hingegen streben sie nach Null. Ferner erkennt man, daß $f_n(\pm 1) = \frac{1}{2}$ für alle $n \in \mathbb{N}$ gilt. Wir vermuten daher, daß die Funktionenfolge im Sinne der vorhin

angedeuteten Konvergenz gegen die *Grenzfunktion*

$$f(x) = \begin{cases} 1, & |x| > 1 \\ \frac{1}{2}, & |x| = 1 \\ 0, & |x| < 1 \end{cases}$$

strebt. Diese Vermutung läßt sich leicht beweisen: Mit Hilfe der Umformung

$$f_n(x) = 1 - \frac{1}{1 + x^{2n}}$$

folgt die Behauptung unter Verwendung der elementaren Grenzwerte

$$\lim_{n\to\infty} x^n = 0 \Leftrightarrow |x| < 1$$
$$\lim_{n\to\infty} x^n = \infty \Leftrightarrow x > 1.$$

Aus Abb. 2.6 kann man ablesen, daß $\lim_{n\to\infty} g_n(x) = 0$ für jedes $x \geq 0$ gilt. Dies ist trivial für $x = 0$. Für $x > 0$ stützt sich der Beweis auf die Ungleichung

$$e^{-x} \leq \frac{k!}{x^k}, \quad \text{für jedes } k \in \mathbb{N}.$$

Somit folgt (mit $k = 1$) für $x > 0$

$$|g_n(x)| = \sqrt{n}xe^{-nx} \leq \sqrt{n}x\frac{1}{nx} = \frac{1}{\sqrt{n}} \underset{n\to\infty}{\to} 0. \tag{2.2}$$

Die Folge (g_n) konvergiert also gegen die Grenzfunktion $g \equiv 0$. Diese Abschätzung, aber auch Abbildung 2.6 zeigen, daß die Funktionen g_n gleichmäßig klein werden: Legt man um die Grenzfunktion einen "ε–Schlauch für ein beliebiges $\varepsilon > 0$, zeichnet man also auch die Funktionen $g \pm \varepsilon$ ein, liegen alle Funktionen g_n mit $n \geq \frac{1}{\varepsilon}^2$ in dem Streifen zwischen $g - \varepsilon$ und $g + \varepsilon$. Wählt man etwa $\varepsilon = \frac{1}{10}$, so zeigt Abb. 2.7 (zusammen mit geeigneten Monotonieuntersuchungen), daß dies tatsächlich bereits für $n \geq 14$ der Fall ist. Die Diskrepanz zu (2.2) ist auf die dort vorgenommene grobe Abschätzung zurückzuführen.

Man macht sich leicht klar, daß keine der Funktionen f_n zur Gänze in einem ε-Schlauch mit $\varepsilon < \frac{1}{4}$ liegen kann. Man beachte auch, daß sich die Stetigkeit der Funktionen f_n nicht auf die Grenzfunktion f überträgt. Offensichtlich hat die Folge (g_n) bessere Konvergenzeigenschaften als die Folge (f_n). Wir präzisieren nun den qualitativen Unterschied im Verhalten der beiden Folgen:

DEFINITION 2.6. *Es seien $f_n, f\colon I \to \mathbb{K}$.*

1. *Die Folge (f_n)* **konvergiert punktweise** *gegen die Grenzfunktion f genau dann, wenn*

$$\lim_{n\to\infty} f_n(x) = f(x)$$

für alle $x \in I$ gilt.

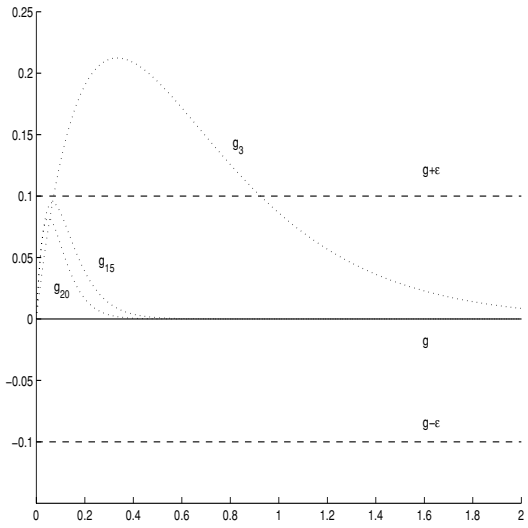


ABB. 2.7. ε-Schlauch, $\varepsilon = \frac{1}{10}$

2. *Die Folge* (f_n) **konvergiert gleichmäßig auf** $\mathbf{D} \subset \mathbf{I}$ *gegen die Grenzfunktion* f *genau dann, wenn* (f_n) *punktweise gegen* f *konvergiert und wenn zu jedem* $\varepsilon > 0$ *ein Index* $N(\varepsilon)$ *gefunden werden kann, sodaß für alle* $x \in D$ *und für alle* $n \geq N(\varepsilon)$

$$|f(x) - f_n(x)| < \varepsilon$$

zutrifft. (Man beachte, $N(\varepsilon)$ *hängt nicht von* x *ab).*
3. *Schreibweise:* $\lim_{n\to\infty} f_n = f$ *pw. bzw.* $\lim_{n\to\infty} f_n = f$ *glm.*

Die Folge (f_n) aus Beispiel 2.5 konvergiert punktweise, die Folge (g_n) gleichmäßig. Wir notieren eine Folgerung, die sich aus der der gleichmäßigen Konvergenz spezieller Funktionenfolgen ergibt:

THEOREM 2.4. *Die Grenzfunktion einer gleichmäßig konvergenten Folge stetiger Funktionen ist stetig.*

2.4. Reihen. Ein wichtiges Beispiel für Folgen sind unendliche Reihen:

DEFINITION 2.7. *Es sei* $(a_n) \subset \mathbb{K}$ *eine Folge reeller oder komplexer Zahlen.*

1. *Für jedes* $n \in \mathbb{N}$ *definiert man*

$$s_n = \sum_{i=1}^{n} a_i$$

und nennt s_n **n-te Partialsumme**.
2. *Die Folge der Partialsummen* (s_n) *heißt* **unendliche Reihe** $\sum_{i=1}^{\infty} a_i$.
3. *Die unendliche Reihe* $\sum_{i=1}^{\infty} a_i$ *heißt* **konvergent** *genau dann, wenn die Folge der Partialsummen konvergent ist.*

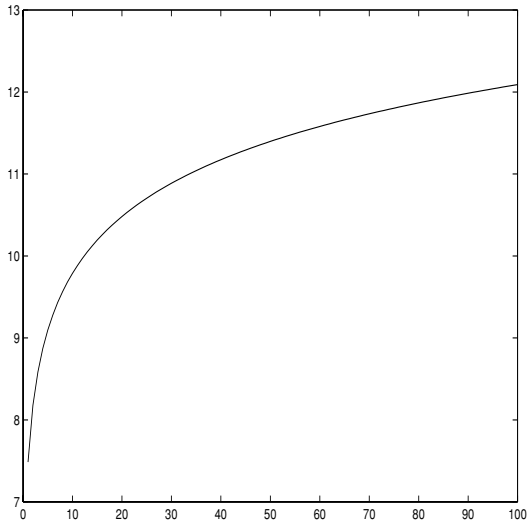


ABB. 2.8. Harmonische Reihe

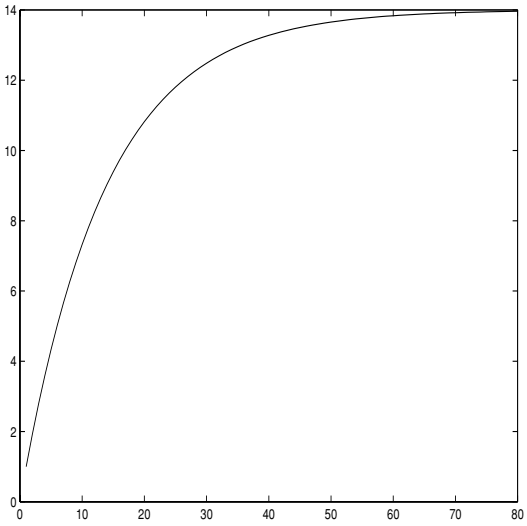


ABB. 2.9. Geometrische Reihe

4. *Schreibweise:* $\sum_{i=1}^{\infty} a_i = \lim_{n\to\infty} s_n$.

Man beachte, daß das Symbol $\sum_{i=1}^{\infty} a_i$ zweifach verwendet wird: Es bezeichnet sowohl die unendliche Reihe selbst als auch im Fall der Konvergenz deren Grenzwert. Für die Untersuchung der Konvergenz einer unendlichen Reihe stehen zahlreiche leistungsfähige Kriterien zur Verfügung. Die analytische Berechnung des Grenzwertes ist jedoch nur in Ausnahmefällen möglich. Meist ist man auf eine numerische Approximation angewiesen.

THEOREM 2.5 (Quotientenkriterium, Wurzelkriterium). *Für die unendlichen Reihe* $\sum_{i=1}^{\infty} a_i$ *existiere mindestens einer der beiden Grenzwerte* $\rho = \lim_{n\to\infty} \frac{|a_{n+1}|}{|a_n|}$ *oder* $\sigma = \lim_{n\to\infty} \sqrt[n]{|a_n|}$

1. *Ist* $\rho < 1$ *oder* $\sigma < 1$, *dann ist die unendliche Reihe konvergent*
2. *Ist* $\rho > 1$ *oder* $\sigma > 1$, *dann ist die unendliche Reihe divergent*
3. *Ist* $\rho = 1$, *dann ist mit Hilfe des Quotientenkriteriums keine Aussage über das Konvergenzverhalten möglich*
4. *Ist* $\sigma = 1$, *dann ist weder mit Hilfe des Wurzelkriteriums noch mit Hilfe des Quotientenkriteriums eine Aussage über das Konvergenzverhalten möglich*

THEOREM 2.6 (Leibnizkriterium). *Eine* **alternierende Reihe**, *also eine Reihe der Form* $\sum_{i=1}^{\infty}(-1)^{i-1}a_i$, $a_i \geq 0$ *für alle* $i \in \mathbb{N}$, *ist konvergent, falls folgende Bedingungen erfüllt sind:*

1. $\lim_{i\to\infty} a_i = 0$,
2. $a_{i+1} \leq a_i$, *für alle* $i \in \mathbb{N}$.

Die Abbildungen (2.8) und (2.9) verdeutlichen, daß auf einen Konvergenztest nicht verzichtet werden kann: Abbildung (2.8) zeigt die Partialsummen s_n der harmonischen Reihe $\sum_{i=1\to\infty}^{\infty} \frac{1}{i}$ für $n = 1000 : 1000 : 100000$, Abb. (2.9) stellt die ersten 80 Partialsummen der geometrischen Reihe $\sum_{i=1}^{\infty}(\frac{13}{14})^i$ dar. Die Abbildungen suggerieren, daß beide Reihen gegen einen Wert in der Nähe von 13 konvergieren. Man kann jedoch *beweisen*, daß die harmonische Reihe tatsächlich divergent ist. Allerdings ist es nicht möglich, die Divergenz dieser Reihe am Rechner zu simulieren. Die Konvergenzvermutung für die geometrische Reihe hingegen kann durch das Wurzelkriterium abgesichert werden.

Die praktische Berechnung des Wertes einer *konvergenten* Reihe wird dadurch erschwert, daß oft nicht klar ist, wieviele Terme der Reihe zu berücksichtigen sind, sodaß der Abbruchfehler eine vorgegebene Toleranz nicht überschreitet.

Erfolgt der Konvergenznachweis mit dem Quotientenkriterium, kann man für den Reihenrest

$$R_n = \sum_{i=n+1}^{\infty} a_i$$

folgende a priori Abschätzung heranziehen:

THEOREM 2.7 (Vergleich mit der geometrischen Reihe). *Gilt für die Glieder der unendlichen Reihe $\sum_{i=1}^{\infty} a_i$ die Abschätzung*

$$|a_{i+1}| \leq \gamma |a_i| \quad \textit{für alle } i \geq n_0$$

mit $0 < \gamma < 1$, dann ist der Reihenrest R_{n_0} beschränkt durch

$$|R_{n_0}| \leq \frac{|a_{n_0+1}|}{1-\gamma}.$$

Ist γ nahe bei 1, kann diese Abschätzung sehr konservativ sein. Ein ähnliches Resultat basiert auf dem Integralkriterium:

THEOREM 2.8 (Vergleich mit einem Integral). *Es sei $f\colon [n_0, \infty) \to \mathbb{R}^+$, $n_0 \in \mathbb{N}$, monoton fallend und es gelte $|a_i| \leq f(i)$ für alle $i \geq n_0$. Dann ist auch*

$$|R_{n_0}| \leq \int_{n_o}^{\infty} f(x)\, dx.$$

Das Restglied kann besonders einfach für eine alternierende Reihe abgeschätzt werden:

THEOREM 2.9. *Das n-te Restglied R_n einer alternierenden Reihe $\sum_{i=1}^{\infty}(-1)^{i-1}a_i$ kann abgeschätzt werden durch*

$$|R_n| \leq a_{n+1}.$$

Der Betrag des n-ten Restgliedes einer alternierenden Reihe ist also nicht größer als der Betrag des ersten venachlässigten Gliedes der Reihe.

BEISPIEL 2.6. Die Eulersche Zahl kann man auch über die *Exponentialreihe* berechnen:

$$e = \sum_{i=0}^{\infty} \frac{1}{i!}$$

Wir wollen diese Darstellung benutzen, um die Eulersche Zahl bis auf 6 Stellen hinter dem Komma genau zu bestimmen.

Die Konvergenz dieser Reihe kann leicht mit dem Quotientenkriterium aus Theorem 2.5 nachgewiesen werden:

$$\frac{|a_{i+1}|}{|a_i|} = \frac{i!}{(i+1)!} = \frac{1}{i+1} \underset{i\to\infty}{\to} 0$$

Daß der Grenzwert dieser Reihe tatsächlich mit dem Grenzwert aus Beispiel 2.1 übereinstimmt, ist nicht trivial und wird in der Analysis bewiesen. Wegen $\rho = 0$ kann man γ in $(0, 1)$ beliebig wählen (warum?), etwa $\gamma = \frac{1}{4}$. Dann ist die Bedingung $|a_{i+1}| \leq \gamma |a_i|$ in Theorem 2.7 erfüllt für $i \geq 3$. Satz 2.7 stellt somit die a priori Abschätzung für den Abbruchfehler R_n

$$|R_n| \leq \frac{|a_{n+1}|}{1-\gamma} = \frac{4}{3}\frac{1}{(n+1)!}$$

für $n \geq 3$ zur Verfügung. Setzt man $tol = 10^{-6}$ und summiert man die Exponentialreihe auf, bis die *Abbruchbedingung*

$$\frac{|a_{n+1}|}{1-\gamma} = \frac{4}{3}\frac{1}{(n+1)!} < tol$$

erfüllt ist, erhält man eine Approximation an die Eulersche Zahl mit der geforderten Genauigkeit. In diesem speziellen Fall könnte man zwar die Abbruchbedingung benützen, um die Anzahl der benötigten Summanden der Exponentialreihe zu berechnen und die Summation mit einer for-Schleife zu realisieren. Einfacher und eleganter lassen sich derartige Aufgabenstellungen —Durchlaufen einer Anweisungssequenz bis eine Abbruchbedingung erfüllt ist— mit einer **while-Schleife** lösen:

```
while (Abbruchbedingung)
        ⋮
   Anweisungen
        ⋮
      end
```

Der while-Block, das sind die Anweisungen zwischen "while" und "end", wird durchlaufen, solange die Abbruchbedingung *erfüllt* ist. Bei der Kodierung der Abbruchbedingung können in MATLAB relationale und logische Operatoren eingesetzt

werden:

Relationale Operatoren		Logische Operatoren	
<	kleiner	&	und
<=	kleiner oder gleich	|	oder
>	größer	~	non
>=	größer oder gleich	xor	ausschließendes oder
==	gleich (=)		
~=	nicht gleich ($\neq$)		

Diese Operatoren haben die gleiche Bedeutung wie in der Mathematik bzw. Logik. Ausführlichere Informationen, etwa über deren Wirkung, falls die Operanden Matrizen sind, findet man in der online Hilfe von MATLAB.

Beispiel 2.7 (Fortsetzung von Beispiel 2.6). Bei der Implementierung des Beispiels sollen nun auch sämtliche Zwischensummen abgespeichert werden. Dies soll die Fähigkeit von MATLAB illustrieren, Felder dynamisch zu dimensionieren.

```
tol= 1e-6;sum = 2;
Sum=[1;sum];                    auf Sum werden die Partialsummen abgespeichert
s=1;                            Hilfsvariable zum Aufbau der Faktoriellen
i=1;
while abs(4*s/(3*(i+1))) > tol  Man achte auf die Ungleichung
  i = i+1;
  s = s/i;                      rekursive Berechnung von 1/i!
  sum = sum + s;
  Sum = [Sum;sum];
end
```

Tabelle2.2 veranschaulicht die rasche Konvergenz der Exponentialreihe. Man vergleiche dies mit den einzelnen Folgengliedern in Beispiel 2.1.

3. Ableitung und Satz von Taylor

Erinnern wir uns zunächst an die aus der Mittelschule geläufige Definition der Ableitung einer Funktion:

Definition 3.1. *Es sei $f\colon I \to \mathbb{R}$ und $x_0 \in I$. Die Abbildung f ist* **differenzierbar in $\mathbf{x_0}$** *genau dann, wenn der Grenzwert des* Differenzenquotienten

$$\lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0}$$

existiert. Dieser Grenzwert heißt **Ableitung von f an der Stelle $\mathbf{x_0}$** *und wird mit $f'(x_0)$ oder $\frac{df}{dx}(x_0)$ bezeichnet. Existiert dieser Grenzwert für alle $x \in I$, heißt f*

i	i-te Partialsume	$\lvert s_i - e\rvert$
0	1.0000000	-1.7182818
1	2.0000000	-0.7182818
2	2.5000000	-0.2182818
3	2.6666667	-0.0516151
4	2.7083333	-0.0099485
5	2.7166667	-0.0016152
6	2.7180556	-0.0002263
7	2.7182540	-0.0000279
8	2.7182788	-0.0000031
9	2.7182815	-0.0000003

TABELLE 2.2. Partialsummen der Exponentialreihe

differenzierbar auf I. *In diesem Falle nennt man die Abbildung*

$$f' \colon \begin{cases} I \to \mathbb{R} \\ x \mapsto f'(x) \end{cases}$$

Ableitung von f.

Ist eine Funktion differenzierbar, kann man die Existenz des Grenzwertes aus Definition 3.1 für f' prüfen und gegebenenfalls die Ableitung von f' untersuchen. Dies führt zur zweiten Ableitung $f'' = (f')'$. Existiert die zweite Ableitung auf I, kann man die dritte Ableitung $f''' = (f'')'$ untersuchen, usw. Die höheren Ableitungen werden demnach rekursiv definiert:

$$\begin{aligned} f^{(0)} &= f \\ f^{(k)} &= (f^{(k-1)})' \quad \text{für } k \in \mathbb{N}. \end{aligned}$$

Anstelle von $f^{(k)}$ ist auch die Schreibweise $\frac{d^k f}{dx^k}$ gebräuchlich.

Anschaulich bedeutet der Differenzenquotient von f die *mittlere Änderungsrate* von f über dem Intervall $[x_0, x]$, die Ableitung $f'(x_0)$ hingegen die *momentane Änderungsrate* an der Stelle x_0. Geometrisch kann man sich den Differenzenquotienten vorstellen als den Anstieg der Sekante durch die Punkte $(x, f(x)$ und $(x_0, f(x_0))$ des Graphen von f. Demnach bedeutet Differenzierbarkeit von f an x_0 gometrisch, daß jede Folge von Sekanten durch den Punkt $(x_0, f(x_0))$ im Grenzwert für $x \to x_0$ einer wohldefinierten Grenzgeraden mit dem Anstieg $f'(x_0)$ zustrebt. Diese Gerade heißt **Tangente** an den Graph von f in x_0 und ihre Gleichung ist

$$t_1(x) = f(x_0) + f'(x_0)(x - x_0). \tag{3.1}$$

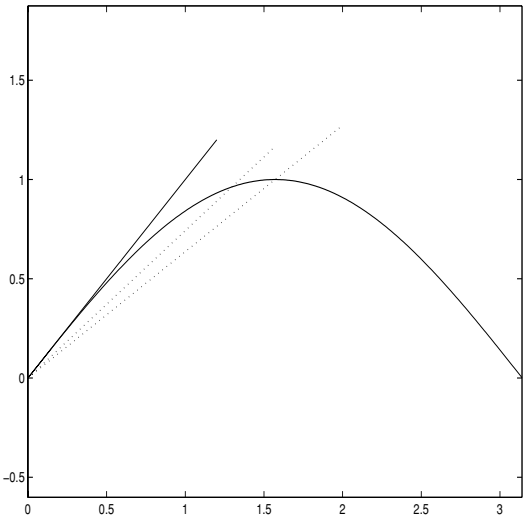


ABB. 3.1. Sekanten und Tangente beim Sinus

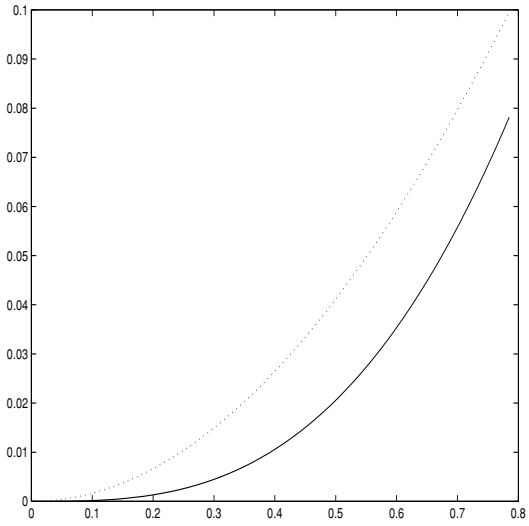


ABB. 3.2. Approximationsfehler $x - sinx$

In Abb. 3.1 werden Sekanten an den Sinus durch die Punkte $(0, \sin 0)$ und $(\frac{\pi}{2}, \sin \frac{\pi}{2})$ bzw. $(\frac{\pi}{4}, \sin \frac{\pi}{4})$ und die Tangente an den Sinus in $x = 0$ dargestellt. Offensichtlich wird durch die Tangente der Graph von f *lokal*, d.h. in einer hinreichend kleinen Umgebung von 0 sehr gut approximiert. Abb. 3.2 zeigt den Approximationsfehler $x-sinx$ (durchgezogen) und $\frac{x-sinx}{x}$ (punktiert). Man erkennt, daß bei Annäherungvon x an $x_0 = 0$ der Approximationsfehler so rasch abklingt daß offensichtlich sogar noch der Quotient $\frac{x-sinx}{x}$ nach 0 strebt.

Man kann die Approximationseigenschaft der Tangente auch für Überschlagsrechnungen nutzen:

BEISPIEL 3.1. Es soll ohne die Hilfe eines Taschenrechners $\sqrt[4]{15}$ berechnet werden. Dazu betrachtet man die auf $(0, \infty)$ differenzierbare Funktion $f(x) = \sqrt[4]{x}$ und beachtet, daß $\sqrt[4]{16} = 2$ gilt. Setzt man also $x = 15$ und $x_0 = 16$ in (3.1), erhält man mit $f'(x) = \frac{1}{4}x^{-\frac{3}{4}}$

$$\sqrt[4]{15} \approx t_1(15) = f(16) + f'(16)(15 - 16) = 2 - \frac{1}{4}\frac{1}{16^{\frac{3}{4}}} = 2 - \frac{1}{32} = 1.96875.$$

Tatsächlich gilt $\sqrt[4]{15} = 1.96799$ (gerundet auf 5 Nachkommastellen). Der Fehler ist somit kleiner als $7,7 \cdot 10^{-4}$. Bei Anwendung dieser Überschlagsrechnung sollte die Stelle x_0 so gewählt werden, daß $f(x_0)$ und $f'(x_0)$ leicht ausgewertet werden können.

Abb. 3.1 legt nahe, daß die Approximation des Sinus in einer Umgebung von $x_0 = 0$ verbessert werden kann, wenn man für die Approximation nicht eine affin lineare Funktion (eine Gerade) sondern ein Polynom 2., 3. etc. Grades verwendet.

Macht man einen quadratischen Ansatz

$$t_2(x) = a_0 + a_1 x + a_2 x^2,$$

benötigt man 3 Gleichungen für die unbekannten Koeffizienten von t_2. Da t_2 die Approximation von f in x_0 verbessern soll und t_1 in x_0 die Bedingungen

$$t_1(x_0) = f(x_0), \quad t_1'(x_0) = f'(x_0)$$

erfüllt, ist es naheliegend, dieselben Bedingungen auch für t_2 und zusätzlich $t''(x_0) = f''(x_0)$ zu verlangen. Für den Sinus ergibt dies das Gleichungssystem:

$$\begin{aligned} t_2(0) &= sin(0) &&\Leftrightarrow \quad a_0 = 0 \\ t_2'(0) &= \tfrac{d}{dx} sin(0) &&\Leftrightarrow \quad a_1 = 1 \\ t_2''(0) &= \tfrac{d^2}{dx^2} sin(0) &&\Leftrightarrow \quad a_2 = 0. \end{aligned}$$

Wir erhalten so das überraschende Resultat, daß auf diese Weise die mit t_1 verfügbare Approximation mit einem quadratischen Ansatz nicht verbessert werden kann:

$$t_2 \equiv t_1.$$

Setzt man Polynome höheren Grades an, erhält man auf ähnliche Weise (vgl. Abb. 3.3)

$$\begin{aligned} t_3(x) &= x - \frac{x^3}{3!}, \\ t_4 &\equiv t_3, \\ t_5(x) &= x - \frac{x^3}{3!} + \frac{x^5}{5!}, \\ &\text{etc.} \end{aligned}$$

Es ist nicht schwer zu zeigen, daß das Polynom n-ten Grades, welches den Bedingungen

$$t_n^{(k)}(x_0) = f^{(k)}(x_0), \quad k = 0, 1, \ldots, n$$

genügt, gegeben ist durch

$$t_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k. \tag{3.2}$$

Das Polynom (3.2) heißt **n-tes Taylorpolynom**. Einer der zentralen Sätze der Analysis beschäftigt sich mit dem Fehler, der entsteht, wenn man eine Funktion f durch das im Allgemeinen einfachere n-te Taylorpolynom ersetzt:

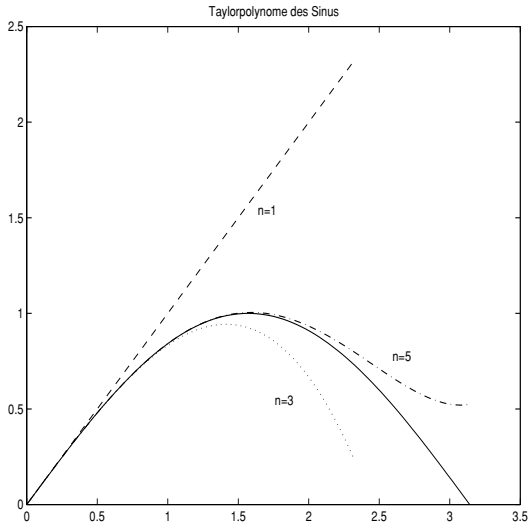


ABB. 3.3. Taylorpolynome des Sinus

THEOREM 3.1 (Satz von Taylor). *Die Funktion $f\colon I \to \mathbb{R}$ sei $n+1$ mal differenzierbar und $x, x_0 \in I$. Dann gibt es zwischen x_0 und x eine Stelle ξ, sodaß die Darstellung*

$$\begin{aligned} f(x) &= t_n(x) + R_n(x) \\ &= \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x-x_0)^k + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1} \end{aligned}$$

gilt.

Man beachte, daß der Fehlerterm dieselbe Struktur hat, wie die Terme des Taylorpolynoms. Allerdings ist über die Stelle ξ nur $x_0 < \xi < x$ (bzw. $x < \xi < x_0$ falls $x_0 < x$) bekannt. Bei der Auswertung des Fehlertermes ist man daher auf Abschätzungen von $f^{(n+1)}$ auf dem Intervall (x_0, x), bzw. (x, x_0) angewiesen. Es ist klar, daß die Lage der Zwischenstelle von n, x_0 und x abhängt. Als Spezialfall (n=0) des Satzes von Taylor notieren wir den Mittelwertsatz:

THEOREM 3.2 (Mittelwertsatz). *Es sei $f\colon [a,b] \to \mathbb{R}$ stetig und differenzierbar auf (a,b). Dann gibt es eine Zwischenstelle $\xi \in (a,b)$ mit*

$$f(b) - f(a) = f'(\xi)(b-a).$$

Sehr viele Funktionen der angewandten Mathematik sind **glatt**, d.h. sie besitzen sämtliche höheren Ableitungen. Für viele von diesen kann man

$$\lim_{n\to\infty} R_n(x) = 0 \tag{3.3}$$

für alle x in einer Umgebung von x_0 nachweisen. In diesem Fall kann man also f beliebig genau durch Taylorpolynome approximieren, soferne nur n hinreichend groß gewählt wird. Als Grenzwert für $n \to \infty$ erhält man die **Taylorreihe**

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k.$$

Taylorreihen sind Spezialfälle von **Potenzreihen,**

$$\mathcal{P}(x) = \sum_{k=0}^{\infty} a_k(x - x_0)^k, \tag{3.4}$$

wobei die **Koeffizienten** a_k, sowie x und x_0 reelle oder komplexe Zahlen sind. Man nennt x_0 das Entwicklungszentrum der Reihe. Zuerst ist natürlich zu klären, für welche x eine Potenzreihe konvergiert. Anwendung des Quotientenkriteriums führt auf

$$\frac{|a_{i+1}(x - x_0)^{i+1}|}{|a_i(x - x_0)^i|} = \frac{|a_{i+1}|}{|a_i|}|x - x_0|.$$

Existiert also

$$\rho = \lim_{i \to \infty} \frac{|a_{i+1}|}{|a_i|}$$

und gilt

$$\rho|x - x_0| < 1,$$

dann konvergiert die Reihe, während im Fall

$$\rho|x - x_0| > 1.$$

Divergenz vorliegt. Allgemein gilt das bemerkenswerte Resultat, daß eine Potenzreihe im Falle $\rho = 0$ auf ganz $\mathbb{R}$ konvergiert, im Falle $0 < \rho < \infty$ auf $(x_0 - \frac{1}{\rho}, x_0 + \frac{1}{\rho})$ konvergiert und außerhalb des Intervalles $[x_0 - \frac{1}{\rho}, x_0 + \frac{1}{\rho}]$ divergiert. Ist $\rho = \infty$ konvergiert die Reihe nur für $x = x_0$. Diesem pathologischen Fall wollen wir keine weitere Aufmerksamkeit schenken. Man nennt $R = \frac{1}{\rho}$ **Konvergenzradius** der Potenzreihe. Jeder Potenzreihe kann somit auf ihrem **Konvergenzintervall** die **Summenfunktion** zugeordnet werden:

$$f(x) = \begin{cases} (x_0 - \frac{1}{\rho}, x_0 + \frac{1}{\rho}) \to \mathbb{R} \\ x \mapsto \sum_{k=0}^{\infty} a_k(x - x_0)^k \end{cases}$$

Man sagt, die Funktion f wird auf dem Konvergenzintervall durch die Potenzreihe dargestellt. Allerdings ist es nur in Ausnahmefällen möglich, einen geschlossenen analytischen Ausdruck für die Summenfunktion einer Potenzreihe zu bestimmen. Die Taylorreihe der Summenfunktion einer Potenzreihe ist die Potenzreihe selbst. Viele Funktionen etwa in der mathematischen Physik sind durch Potenzreihen definiert. Umgekehrt ist man aber oft daran interessiert, eine gegebene Funktion um eine Stelle

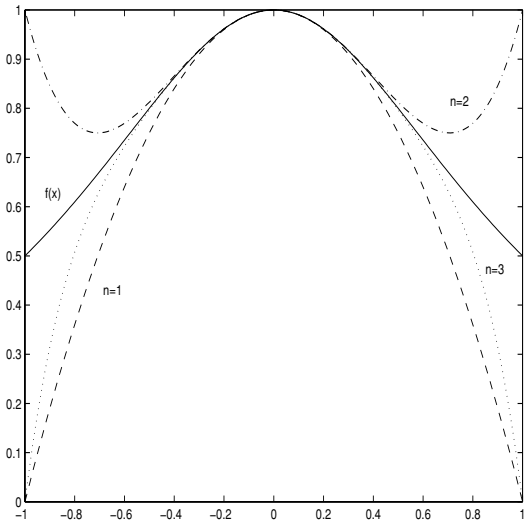


ABB. 3.4. Taylorpolynome für $\frac{1}{1+x^2}$

x_0 in eine Potenzreihe zu entwickeln, denn Potenzreihen bieten eine sehr bequeme, manchmal sogar die einzige Möglichkeit, f numerisch auszuwerten. Eine systematische, aber oft nicht besonders rationelle Vorgangsweise besteht darin, die Funktion in eine Taylorreihe um x_0 zu entwickeln. In Tabelle 3.1 werden einige nützliche Reihenentwicklungen zusammen mit ihren Konvergenzintervallen zusammengestellt.

BEISPIEL 3.2. Entwickelt man die Funktion $f(x) = \frac{1}{1+x^2}$ in eine Potenzreihe um $x_0 = 0$, erhält man $f(x) = \sum_{i=0}^{\infty}(-1)^i x^{2i}$ auf dem Konvergenzintervall $(-1, 1)$. In Abb. 3.4 sind die Taylorpolynome für $n = 1, 2, 3$ dargestellt. Man erkennt deutlich die schlechte Approximation in der Nähe von $x = \pm 1$, dem Rand des Konvergenzintervalles. Da die Taylorpolynome (also die Partialsummen der Potenzreihe) mit

TABELLE 3.1. Potenzreihenentwicklungen

$f(x)$	Potenzreihe	konvergiert
e^x	$1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$	$\mathbb{R}$
$\sin x$	$x - \frac{x^3}{3!} + \frac{x^5}{5!} \mp \cdots$	$\mathbb{R}$
$\cos x$	$1 - \frac{x^2}{2!} + \frac{x^4}{4!} \mp \cdots$	$\mathbb{R}$
$\ln(1+x)$	$x - \frac{x^2}{2} + \frac{x^3}{3} \mp \cdots$	$(-1, 1]$
$\frac{1}{1-x}$	$1 + x + x^2 + x^3 + \cdots$	$(-1, 1)$
$\arctan x$	$x - \frac{x^3}{3} + \frac{x^5}{5}$	$(-1, 1]$

einer geraden Anzahl von Summanden in ± 1 stets Null sind, jene mit einer ungeraden Anzahl von Summanden dort stets den Wert 1 annehmen, ist die Potenzreihe in den Randpunkten des Konvergenzintervalles divergent. Die Funktion f ist aber auf $\mathbb{R}$ beliebig oft differenzierbar. Es ist daher nicht einsichtig, daß die Entwicklung dieser Funktion um $x_0 = 0$ auf eine Reihe mit Konvergenzradius 1 führt. Dies wird erst dann verständlich, wenn man f ins Komplexe fortsetzt.

Wieviele Terme der Reihenentwicklung sind zu berücksichtigen, damit der Approximationsfehler auf dem Intervall $[-\frac{3}{4}, \frac{3}{4}]$ nicht größer als 10^{-5} ist? Da die Potenzreihe eine alternierende Reihe darstellt, kann der Betrag des Reihenrestes durch den Betrag des ersten vernachlässigten Reihengliedes abgeschätzt werden:

$$x^{2n} \leq (\frac{3}{4})^{2n} \leq 10^{-5}.$$

Bildet man den dekadischen Logarithmus dieser Ungleichung, erhält man die Bedingung

$$n \geq -\frac{5}{2 \log \frac{3}{4}} > 20.0098.$$

Man muß also die ersten 21 Terme in der Reihe berücksichtigen, um die geforderte Genauigkeit für $|x| \leq \frac{3}{4}$ zu gewährleisten.

4. Unterprogramm Technik in MATLAB

Sehr oft hängt der weitere Ablauf eines Programmes von einer oder mehreren Bedingungen ab: Je nachdem welche Bedingung erfüllt ist, soll eine bestimmte Aktion ausgeführt werden. Eine derartige bedingte Programmverzweigung kann in MATLAB mit folgender Struktur implementiert werden:

if Bedingung 1
⋮
Alternative 1
⋮
elseif Bedingung 2
⋮
Alternative 2
⋮
⋮
elseif Bedingung n
⋮
Alternative n
⋮
else
⋮

Alternative
⋮
end
Anweisungen

Diese Sequenz wird folgendermaßen abgearbeitet: Zuerst wird die Bedingung 1 getestet. Ist sie erfüllt, werden die Anweisungen der Alternative 1 ausgeführt und danach die Anweisungen nach "end". Ist die Bedingung 1 nicht erfüllt, wird die Bedingung 2 abgefragt und genauso wie vorhin verfahren. Sind alle "elseif"–Bedingungen nicht erfüllt, wird die Alternative nach "else" ausgeführt und anschließend das Programm mit den Anweisungen nach "end" fortgesetzt. Es können mehrere "elseif" Abfragen verwendet werden, "else" darf naturgemäß höchstens einmal eingesetzt werden. Sowohl "elseif", als auch "else" Blöcke sind optional. Die Verzweigungsbedingungen sind wie bei der "while"–Schleife aufgebaut. Häufig auftretende Spezialfälle der bedingten Programmverzweigung sind:

if Bedingung	if Bedingung
⋮	⋮
Anweisungen	Anweisungen 1
⋮	⋮
end	else
	⋮
	Anweisungen 2
	⋮
	end

Beispiel 4.1. Dieses Beispiel ist der online help Information zur "if" Anweisung entnommen. Man beachte allerdings, daß das Programmfragment in dieser Form (noch) nicht exekutierbar ist. Welche Ergänzungen müßte man machen, um ein lauffähiges Programm zu erhalten?

```
if i == j              == logische Gleichheit
   A(i,j) = 2;
elseif abs(i-j) == 1
   A(i,j) = -1;        = Wertzuweisung
else
   A(i,j) = 0;         Die Einrückung erleichtert nur die Lesbarkeit
end
```

Man beachte, daß die Abfrage auf Gleichheit nur für ganze Zahlen sinnvoll ist. Offensichtlich soll eine strukturierte Matrix aufgebaut werden: Wenn Zeilen– und Spaltenindex eines Matrixelementes gleich sind (Bedingung 1) wird ihm der Wert 2 zugewiesen (Alternative 1), wenn sich Zeilen– und Spaltenindex um genau 1 unterscheiden (Bedingung 2), hat das entsprechende Matrixelement den Wert -1 (Alternative 2), allen anderen Matrixelementen wird der Wert 0 zugewiesen (Alternative).

Diese Matrix tritt z.B. bei der Diskretisierung von Differentialgleichungen 2. Ordnung auf.

Bisher haben wir MATLAB nur interaktiv eingesetzt. Für komplexere Programme ist dieser Modus nur bedingt geeignet. Das Suchen eines Befehles im Befehlszeilenspeicher wird überflüssig, wenn man die Anweisungen in einem *Skript* ablegt. Ein Skript ist ein File mit der Erweiterung .m, in den man die Anweisungen so hineinschreibt, als würde man das entsprechende Problem interaktiv mit MATLAB bearbeiten. Um die Integrität des MATLAB Dateisystems nicht zu zerstören, ist es zweckmäßig, ein für alle Mal im Stammverzeichnis ein individuelles Verzeichnis, z.B. "MeineProgramme" anzulegen, und gleich zu Beginn jeder MATLAB Sitzung mit "cd \MeineProgramme" in dieses Verzeichnis zu wechseln. Mit "pwd" (**p**rint **w**orking **d**irectory) kann das jeweilige Arbeitsverzeichnis von MATLAB abgefragt werden. In dieses Verzeichnis wird das Skript unter einem deskriptiven Namen z.B. "programm.m" abgespeichert. Das erste Zeichen des Filenamens muß ein Buchstabe sein, es sollten auch keine Sonderzeichen (+, - , %, ;, etc.) verwendet werden. Weitere Einschränkungen können durch das Betriebssystem verursacht werden. Die Eingabe "programm" (keine Erweiterung!) in der MATLAB Befehlszeile hat nun dieselbe Wirkung, als hätte man die Befehle des Skripts einzeln nacheinander an dieser Stelle eingegeben. Insbesonders beziehen sich gleichnamige Variablen des Workspace und des Skripts auf denselben Speicherbereich. Dies schränkt den komfortablen Anwendungsbereich von Skripts erheblich ein.

Eine saubere Trennung von Variablen eines m-Files und der Variablen des Workspace bietet das Funktionsprogramm. Auch dieses ist ein File mit der Erweiterung ".m", die erste Zeile hat jedoch folgende Syntax:

function outputliste = filename(inputliste)

Der vollständige Name des function–Files ist in diesem Fall "filename.m". Ist das erste Wort eines m-Files "function", erkennt MATLAB, daß eine Funktion (entspricht einem Unterprogramm z.B. in FORTRAN) vorliegt und legt für diese Funktion einen eigenen Workspace an. Alle Variablen der Funktion existieren nur lokal. Gleichnamige Variablen im aufrufenden Workspace und im Workspace der Funktion sind voneinander unabhängig. Die Kommunikation zwischen aufrufendem Workspace und Funktion erfolgt über die Input/Output Listen. Es kann sowohl die Inputliste als auch die Outputliste fehlen. Alle Variablen, welche nicht in der Inputliste stehen, müssen im Unterprogramm initialisiert werden. Dies kann natürlich mit Hilfe der Variablen der Inputliste erfolgen. Auf Variablen, die nicht in der Outputliste stehen, kann außerhalb des Unterprogrammes (in der Bedeutung, welche sie im Unterprogramm hatten) nicht mehr zugegriffen werden. Die Zuordnung der Variablen in der Input bzw. Outputliste erfolgt nicht über den Namen, sondern über die Position in der Liste.

Es ist gute Programmierpraxis gleich von Beginn an, Unterprogramme ausreichend zu kommentieren. Das Kommentarzeichen "%" deutet das logische Ende einer Zeile an, der MATLAB Interpreter/Compiler ignoriert alle nachfolgenden Eingaben in derselben Zeile, also die gesamte Zeile, falls "%" an die erste Position der Zeile gesetzt wird. Existiert z.B. eine Funktion "myprogram.m", kann mit "help myprogram" auf den ersten zusammenhängenden Block von Kommentarzeilen zugegriffen werden, der unmittelbar der Definitionszeile folgt.

Beispiel 4.2. Die Unterprogrammtechnik in MATLAB soll am Beispiel einer Funktion, welche das arithmetische und geometrische Mittel von n Zahlen berechnet, demonstriert werden. Die Daten sind im Vektor x abgespeichert. Das Unterprogramm muß daher zuerst die Anzahl der zu mittelnden Zahlen feststellen.

```
function [am,gm] = mittel(x)   %Definitionszeile
% Arithmetisches und geometrisches Mittel
% Aufruf: [am,gm] = mittel(x)
% Input: x enthält die zu mittelnden Daten
% Output: am ... arithmetisches Mittel
%         gm ... geometrisches Mittel

n = length(x);                 %Bestimmen der Anzahl der Daten
am = sum(x)/n;                 % arithmetisches Mittel
gm = prod(x).^(1/n);           % geometrisches Mittel
```

Existiert im Basisworkspace ein Vektor a = [1 2 3], wird z.B. mit

```
[x,y] = mittel(a);
```

das arithmetische und geometrische Mittel der Zahlen in a berechnet. Eine Funktion kann selbst wieder eine Funktion aufrufen: In diesem Beispiel die eingebauten Funktionen "length", "sum" und "prod", welche die Anzahl, die Summe und das Produkt der Elemente von x berechnen.

Die Funktion "mittel" enthält bis jetzt jedoch einen schweren Fehler (welchen?). Es wird nicht berücksichtigt, daß das geometrische Mittel nur für positive reelle Zahlen definiert ist. Wenn wir annehmen, daß komplexe Zahlen unter den Daten nicht auftreten können, ist das geometrische Mittel trotzdem eine komplexe Zahl, falls das Produkt der Elemente von x negativ ist. Dies wäre bereits ein deutlicher Hinweis auf einen Fehler im Programm. Viel schlimmer ist, daß ohne erkennbare Hinweise ein sinnloses Resultat berechnet wird, wenn eine gerade Anzahl von Daten negativ ist. Wir müssen das Programm also mit einer Abfrage erweitern, ob negative Zahlen in der Inputliste auftreten. Sollte dies der Fall sein, sollte das Programm mit der Ausgabe des nach wie vor sinnvollen arithmetischen Mittels und einer Warnung bezüglich der Nichtexistenz des geometrischen Mittels terminieren:

```
function [am,gm] = mittel(x)                          %Definitionszeile
% Arithmetisches und geometrisches Mittel
% Aufruf: [am,gm] = mittel(x)
%Input: x enthält zu mittelnde Daten
%Output: am ... arithmetisches Mittel
%         gm ... geometrisches Mittel

n = length(x);                                        %Bestimmen der Anzahl der
                                                      Daten
am = sum(x)/n;                                        % arithmetisches Mittel
neg = find(x < 0);                                    %bestimmt die Indizes nega-
                                                      tiver Elemente von x
if isempty(neg)
   gm = prod(x).^(1/n);                               % geometrisches Mittel
else
   disp('Daten mit Indizes '),neg, disp('sind negativ')
   gm = [ ];
end
```

Am interessantesten in dieser Modifikation ist der Befehl "find". In der online Hilfe findet man dazu die Information, daß "find" die Indizes von Vektorelementen findet, welche nicht Null sind. Um die Wirkungsweise von "neg = find(x < 0)" zu verstehen, muß man berücksichtigen, daß das Ergebnis der Abfrage "x < 0" ein Vektor ist, dessen i–tes Element 1 bzw. 0 ist, falls $x(i) < 0$ zutrifft bzw. falsch ist. Zur Anweisung "disp" (für display) sei ergänzend zur online Hilfe bemerkt, daß Textelemente grundsätzlich zwischen einfachen Anführungszeichen zu setzen sind.

KAPITEL 2

Nichtlineare Gleichungen

1. Das Bisektionsverfahren

Das Aufsuchen von Lösungen einer Gleichung

$$f(x) = 0,$$

d.h. von **Nullstellen** von f, wird in nahezu allen Anwendungen notwendig. Eines der einfachsten Verfahren zur Berechnung von (reellen) Nullstellen einer *stetigen* Funktion beruht auf der Zwischenwerteigenschaft 2.2. Es ist meist relativ leicht, Stellen $a < b$ mit

$$f(a)f(b) < 0$$

zu finden. Die Funktionswerte von f in a und b haben dann verschiedenes Vorzeichen. Der Zwischenwertsatz 2.2 sichert daher die Existenz einer Nullstelle von f in (a, b). Man halbiert nun das Intervall $[a, b]$ und behält jene Hälfte, auf der das Vorzeichen von f wechselt. Auf diese Weise fortfahrend, erhält man eine Folge von abgeschlossenen Intervallen, welche jeweils mindestens eine (aber nicht notwendigerweise dieselbe) Nullstelle von f enthalten. Es ist intuitiv klar, daß sich die Intervalle auf eine wohlbestimmte Zahl ξ zusammenziehen, tatsächlich aber verbirgt sich dahinter eine fundamentale Eigenschaft der reellen Zahlen: *Das Prinzip der Intervallschachtelung.* Aus der Stetigkeit von f folgt dann $f(\xi) = 0$.

ALGORITHMUS 1.1 (Bisektionsverfahren). *Es sei $f\colon [a, b] \to \mathbb{R}$ eine stetige Funktion mit $f(a)f(b) < 0$ und $\varepsilon > 0$ eine Fehlertoleranz. Das Bisektionsverfahren besteht aus folgenden Schritten:*

Schritt 1: *Setze $m = \frac{a+b}{2}$*
Schritt 2: *Falls $b - m \leq \varepsilon$ akzeptiere m und stop*
Schritt 3: *Falls $f(m) = 0$ akzeptiere m und stop*
Schritt 4: *Falls $f(m)f(b) < 0$ setze $a = m$*
andernfalls setze $b = m$ und gehe zu Schritt 1

BEISPIEL 1.1. Man bestimme die kleinste positive Lösung der Gleichung

$$e^{-x} = \sin x \tag{1.1}$$

mit einer Genauigkeit von $\varepsilon = 0.001$. Wir setzen $f(x) = e^{-x} - \sin x$, $a = 0$ und $b = \frac{\pi}{2}$. Wegen $f(0) = 1$ und $f(\frac{\pi}{2}) = e^{-\frac{\pi}{2}} - 1 < 0$ ist die Bedingung $f(a)f(b) < 0$ erfüllt. Ein

n	a	b	m	b-m	$f(m)$
1	0.0000	1.5708	0.7854	0.7854	-0.2512
2	0.0000	0.7854	0.3927	0.3927	0.2925
3	0.3927	0.7854	0.5890	0.1963	-0.0007
4	0.3927	0.5890	0.4909	0.0982	0.1407
5	0.4909	0.5890	0.5400	0.0491	0.0687
6	0.5400	0.5890	0.5645	0.0245	0.0336
7	0.5645	0.5890	0.5768	0.0123	0.0164
8	0.5768	0.5890	0.5829	0.0061	0.0078
9	0.5829	0.5890	0.5860	0.0031	0.0035
10	0.5860	0.5890	0.5875	0.0015	0.0014
11	0.5875	0.5890	0.5883	0.0008	0.0003

TABELLE 1.1. Bisektionsverfahren für 1.1

besseres Startintervall könnte natürlich aus dem Graphen von f abgelesen werden. Tabelle 1.1 zeigt den Ablauf der einzelnen Iterationen des Bisektionsalgorithmus.

Fehleranalyse. Im Verfahren wird eine Folge von Intervallen $[a_n, b_n]$ mit

$$b_{n+1} - a_{n+1} = \frac{1}{2}(b_n - a_n), \qquad n \geq 0$$

berechnet. Ein einfaches Induktionsargument zeigt

$$b_n - a_n = 2^{-n}(b - a), \qquad n \geq 1, \tag{1.2}$$

falls $a_0 = a$ und $b_0 = b$ gesetzt wird. Im n–ten Schritt, $n \geq 1$, des Verfahrens steht das Intervall $[a_{n-1}, b_{n-1}]$ zur Verfügung, das eine Nullstelle ξ enthält. Die n–te Approximation m_n ist der Mittelpunkt dieses Intervalles, also gilt

$$|m_n - \xi| \leq \frac{1}{2}(b_{n-1} - a_{n-1}).$$

Mit (1.2) folgt die Fehlerabschätzung

$$|m_n - \xi| \leq \frac{1}{2^n}(b - a). \tag{1.3}$$

Man kann (1.3) benutzen, um die Anzahl der Iterationen abzuschätzen, welche notwendig ist, um eine geforderte Genauigkeit ε zu erreichen: Dies ist sicherlich der Fall, wenn n die Bedingung

$$\frac{1}{2^n}(b - a) < \varepsilon \tag{1.4}$$

erfüllt. Logarithmiert man (1.4) und löst die Ungleichung nach n auf, erhält man

$$n > -\frac{\log \varepsilon - \log(b-a)}{\log 2}. \tag{1.5}$$

Für Beispiel 1.1 ergibt dies bei Verwendung dekadischer Logarithmen und $\varepsilon = 10^{-3}$

$$n > -\frac{-3 - \log \frac{\pi}{2}}{\log 2} > 10.6,$$

in guter Übereinstimmung mit der tatsächlich benötigten Anzahl von Iterationen.

Die Vorzüge des Bisektionsverfahrens sind:

- Garantierte Konvergenz,
- Fehlerabschätzung, (1.3)
- benötigt nur 1 Funktionsauswertung je Iteration,
- benötigt nur Stetigkeit von f.

Dem steht gegenüber, daß das Verfahren relativ langsam konvergiert. Im Mittel braucht man etwas mehr als 3 Iterationen, um die Approximation um eine Dezimalstelle zu verbessern.

2. Das Newton Verfahren

Diese Methode benutzt die Approximationseigenschaft der Ableitung: Angenommen x_0 ist eine gute Näherung für eine Nullstelle ξ. Man ersetzt dann die nichtlineare Funktion f durch das 1. Taylorpolynom (3.1)

$$t_1(x) = f(x_0) + f'(x_0)(x - x_0)$$

und berechnet die Nullstelle x_1 von t_1

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Dies ist möglich falls $f'(x_0) \neq 0$ gilt. Meist wird x_1 näher bei ξ liegen als x_0 und es ist sinnvoll, diese Vorgangsweise mit x_1 an Stelle von x_0 zu wiederholen. Dies führt zu folgendem Algorithmus:

Algorithmus 2.1 (Newtonverfahren). *Es sei $f\colon I \to \mathbb{R}$ eine differenzierbare Funktion und $tol > 0$ eine gegebene Fehlertoleranz.*

Schritt 0: *Wähle Startwert x_0*

Schritt k: *Es steht x_{k-1} zur Verfügung, berechne*

$$x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})}, \tag{2.1}$$

teste Abbruchbedingung

$$|x_{k-1} - x_k| < tol.$$

n	x_n	$f(x_n)$	$x_n - x_{n-1}$
0	1.57079633	$-7.92e-1$	
1	-2.23968105	$1.02e+1$	$-3.81e+0$
2	-1.07952265	$3.83e+0$	$1.16e+0$
3	0.04053023	$9.20e-1$	$1.12e+0$
4	0.50992563	$1.12e-1$	$4.69e-1$
5	0.58623480	$3.19e-3$	$7.63e-2$
6	0.58853063	$2.92e-6$	$2.30e-3$
7	0.58853274	$2.47e-12$	$2.11e-6$
8	0.58853274	$0.00e+000$	$1.78e-12$

TABELLE 2.1. Newton Verfahren zu Beispiel 1.1

Tabelle 2.1 zeigt die Newton Iterationen für Beispiel 1.1 zum Startwert $x_0 = \frac{\pi}{2}$, allerdings mit $tol = 10^{-6}$. Das Verfahren wird abgebrochen, wenn $|x_n - x_{n-1}| < tol$. Wir werden später erläutern, warum dies eine sinnvolle Abbruchbedingung darstellt. Man erkennt, daß eine mit dem Ergebnis des Bisektionsverfahrens (mit $tol = 10^{-3}$!) vergleichbare Approximation bereits nach 6 Iteration erreicht wird. In der Anfangsphase konvergiert das Verfahren noch langsam (wenn überhaupt), die Konvergenz wird dann immer rascher, ab der 8. Iteration stimmen aufeinanderfolgende Iterationen mindestens in den ersten 11 Dezimalstellen überein. Dieses Verhalten ist charakteristisch für das Newton Verfahren.

2.1. Fehleranalyse. Für die folgende Fehleranalyse voraus wir voraus, daß f in einer Umgebung der gesuchten Nullstelle ξ mindestens 2 mal differenzierbar ist, f'' stetig ist, und

$$f'(\xi) \neq 0$$

gilt. Wegen der Stetigkeit von f', welche durch die vorausgesetzte Existenz von f'' gesichert ist, folgt daher $f'(x) \neq 0$ für alle x hinreichend nahe bei ξ. Somit ist die Newtoniteration sinnvoll in einer Umgebung der Nullstelle. Entwickelt man f mit Hilfe des Taylorschen Satzes 3.1 um x_n, erhält man

$$f(x) = f(x_n) + f'(x_n)(x - x_n) + \frac{1}{2} f''(c_n)(x - x_n)^2, \tag{2.2}$$

mit einem nicht näher bekannten $c_n \in I(x_n, x)$. Mit $I(a, b)$ bezeichnen wir ein Intervall mit den Randpunkten a und b. Auswertung von (2.2) an der Stelle $x = x_{n+1}$ ergibt

$$f(x_{n+1}) = f(x_n) + f'(x_n)(x_{n+1} - x_n) + \frac{1}{2} f''(c_n)(x_{n+1} - x_n)^2. \tag{2.3}$$

Wegen der Iterationsvorschrift (2.1) gilt

$$f(x_n) + f'(x_n)(x_{n+1} - x_n) = 0$$

und somit

$$f(x_{n+1}) = \frac{1}{2} f''(c_n)(x_{n+1} - x_n)^2.$$

Subtrahiert man auf der linken Seite $f(\xi)$ (man beachte $f(\xi) = 0$) und wendet auf die Differenz den Mittelwertsatz 3.2 an, ergibt sich auf der linken Seite von (2.3)

$$f(x_{n+1}) - f(\xi) = f'(d_n)(x_{n+1} - \xi)$$

mit $d_n \in I(x_{n+1}, \xi)$. Dividiert man durch $f'(d_n)$ ($\neq 0$ zumindest hinreichend nahe bei der Nullstelle), erhält man

$$\xi - x_{n+1} = -\frac{1}{2} \frac{f''(c_n)}{f'(d_n)} (x_{n+1} - x_n)^2. \tag{2.4}$$

Der Fehler in x_{n+1} ist somit proportional zum Quadrat des Abstandes von x_{n+1} zu x_n. Ein derartiges Verfahren nennt man **quadratisch konvergent** (vgl. Definition 2.3). Diese rasche Konvergenz ist der Grund für die große Beliebtheit des Newtonverfahrens. Für praktische Zwecke ist (2.4) wegen der Unbekannten c_n und d_n noch nicht brauchbar. Es bieten sich zwei Alternativen an: Man kann in (2.4) Absolutbeträge setzen und zu einer *Ungleichung* übergehen,

$$|\xi - x_{n+1}| \leq \frac{1}{2} \frac{\max_{x \in I} |f''(x)|}{\min_{x \in I} |f'(x)|} |x_{n+1} - x_n|^2, \tag{2.5}$$

wobei I für ein Intervall steht, in dem die Iterationen und die Nullstelle ξ liegen. Wie man ein derartiges Intervall *a priori* bestimmen kann, soll hier nicht weiter untersucht werden. Eine andere Modifikation von (2.4) benützt die Stetigkeit von f' und f''. Für jene Iterationen, welche in der Nähe der gesuchten Nullstelle liegen, gilt dann

$$f''(c_n) \approx f''(x_{n+1}), \qquad f'(d_n) \approx f'(x_{n+1}),$$

woraus

$$\xi - x_{n+1} \approx -\frac{1}{2} \frac{f''(x_{n+1})}{f'(x_{n+1})} (x_{n+1} - x_n)^2. \tag{2.6}$$

folgt (das Zeichen "$\approx$" bedeutet ungefähr gleich).

Beispiel 2.1. Wir wollen nun die Fehlerabschätzungen (2.5) und (2.6) auf die letzten Newtoniterationen in Beispiel 1.1 anwenden. Man benötigt

$$f'(x) = -e^{-x} - \cos x$$
$$f''(x) = e^{-x} + \sin x.$$

Wir betrachten nur die Iterationen in $I = [0, 1]$. Auf I ist f' monoton steigend (wegen $f'' > 0$) und $f < 0$, daher folgt

$$\min_{x \in I} |f'(x)| = e^{-1} + \cos 1.$$

Die 2.Ableitung wollen wir nur sehr grob abschätzen durch

$$|f''(x)| = | - e^{-x} + \cos x| \leq 2.$$

Dies ergibt für den Quotienten in (2.5)

$$\frac{1}{2} \frac{\max_{x \in I} |f''(x)|}{\min_{x \in I} |f'(x)|} \leq \frac{1}{e^{-1} + \cos 1} < 1.1012.$$

In Tabelle 2.2 werden die Fehler (2.5) und (2.6) in der n–ten Approximation von ξ einander gegenübergestellt:

n	x_n	Fehler (2.5)	Fehler (2.6)
5	0.50993	0.24261	0.08140
6	0.58623	0.0064118	0.0023252
7	0.58853	$5.804e-6$	$2.100e-6$
8	0.58853	$4.90e-12$	$1.78e-12$
9	0.58853	$3.5e-24$	$1.3e-24$

TABELLE 2.2. Abbruchfehler des Newton Verfahrens

Die Schranke für den Fehler nach (2.6) ist meist etwas kleiner als nach (2.5). Allerdings gibt (2.6) nur eine Vorstellung über die Größenordnung des Fehlers, während durch die Abschätzung (2.5) ein Intervall bestimmt ist, in dem die gesuchte Nullstelle tatsächlich liegt. Dies verdeutlicht nun auch, warum die Abbruchbedingung

$$|x_n - x_{n-1}| < tol \tag{2.7}$$

sinnvoll ist.

Für betragsmäßig große Nullstellen kann ein Kriterium für den relativen Fehler zweckmäßiger sein:

$$|x_n - x_{n-1}| < tol|x_n| \tag{2.8}$$

Setzt man z.B. $tol = 10^{-m}$, terminiert die Bedingung (2.7) das Verfahren erst dann, wenn aufeinanderfolgende Iterierte x_n, x_{n-1} in den ersten m *Nachkomma*stellen übereinstimmen, die bei betragsmäßig großen Zahlen oft von untergeordnetem Interesse sind, während (2.8) bereits abbricht, wenn aufeinander folgende Iterationen in den ersten m Stellen übereinstimmen.

$n+1$	$\frac{x_{n+1}-x_n}{(x_n-x_{n-1})^2}$
3	0.080
4	0.832
5	0.374
6	0.346
7	0.394
8	0.400
9	0.400

TABELLE 2.3. Konvergenzrate des Newton Verfahrens

Die quadratische Konvergenz des Verfahrens kann man dokumentieren, indem man $\frac{x_{n+1}-x_n}{(x_n-x_{n-1})^2}$ berechnet. Analog zu (2.4) gilt nämlich auch

$$x_{n+1} - x_n = \frac{1}{2}\frac{f''(c_{n-1})}{f'(x_n)}(x_n - x_{n-1})^2.$$

Wegen der Stetigkeit der ersten und zweiten Ableitung von f und wegen $\lim_{n\to\infty} x_n = \lim_{n\to\infty} c_n = \xi$ folgt

$$\lim_{n\to\infty} \frac{x_{n+1} - x_n}{(x_n - x_{n-1})^2} = \frac{1}{2}\frac{f''(\xi)}{f'(\xi)}.$$

Wir können diesen Grenzwert zwar nicht a priori auswerten, da wir ξ nicht kennen, aber wir können die Konvergenz dieser Quotientenfolge numerisch überprüfen:

Aus Tabelle 2.3 liest man

$$\lim_{n\to\infty} \frac{x_{n+1} - x_n}{(x_n - x_{n-1})^2} \approx 0.400$$

ab, dies stimmt sehr gut mit $\frac{1}{2}\frac{f''(x_9)}{f'(x_9)}$ überein. Abschließend fassen wir die Vor– und Nachteile des Newton Verfahrens zusammen:

+ quadratische Konvergenz
− in jedem Iterationsschritt muß f und f' ausgewertet werden
− Verfahren muß nicht konvergieren

3. Das Sekantenverfahren

Ein gravierender Nachteil des Newton Verfahrens besteht darin, daß der Rechenaufwand pro Iteration durch die zusätzliche Auswertung von f' grob gesprochen verdoppelt wird. Um den Rechenaufwand zu reduzieren, ohne die Konvergenzgeschwindigkeit des Verfahrens zu sehr zu beeinträchtigen, könnte man daran denken, den Graph von

f nicht durch die *Tangente* in $(x_0, f(x_0))$, sondern durch die *Sekante* durch *zwei* Startwerte $(x_0, f(x_0))$, $(x_1, f(x_1))$,

$$s(x) = f(x_1) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_1)$$

lokal zu approximieren und x_2 aus $s(x) = 0$ zu berechnen. Falls $x_0 \neq x_1$ so gewählt waren, daß $f(x_0) \neq f(x_1)$, ergibt dies

$$x_2 = x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)}.$$

Nun bestimmt man die Sekante durch $(x_1, f(x_1))$, $(x_2, f(x_2))$ (x_0 wird dafür nicht mehr benötigt) und berechnet x_3, etc. Iterativ fortfahrend erhält man folgenden Algorithmus:

ALGORITHMUS 3.1 (Sekantenverfahren). *Es sei $f\colon I \to \mathbb{R}$ eine stetige Funktion und $tol > 0$ eine gegebene Fehlertoleranz.*

Schritt 0: *Bestimme 2 Startwerte x_0, x_1 mit $f(x_0) \neq f(x_1)$*

Schritt k: *Es stehen x_{k-1}, x_{k-2} zur Verfügung*
Berechne x_k aus

$$x_k = x_{k-1} - f(x_{k-1})\frac{x_{k-1} - x_{k-2}}{f(x_{k-1}) - f(x_{k-2})}, \tag{3.1}$$

teste Abbruchbedingung

$$|x_{k-1} - x_k| \leq tol.$$

Die Iterationsvorschrift (3.1) des Sekantenverfahrens ist also der des Newton Verfahrens (2.1) sehr ähnlich: Es wird lediglich die Ableitung $f'(x_n)$ durch den entsprechenden Differenzenquotienten ersetzt. In jedem Iterationsschritt mit Ausnahme des ersten ist nur eine Auswertung von f erforderlich.

BEISPIEL 3.1. Wendet man das Sekantenverfahren an auf $f(x) = e^{-x} - \sin x$ mit den Startwerten $x_0 = 0$ und $x_1 = \frac{\pi}{2}$ und $tol = 10^{-6}$, erhält man die Iterationen in Tabelle 3.1. Daraus geht hervor, daß das Sekantenverfahren eine Nullstelle mit einer dem Newton Verfahren vergleichbaren Anzahl von Iterationen findet (in diesem Beispiel ist die Wahl der Startwerte besonders günstig). Die letzte Spalte allerdings deutet an, daß das Sekanten Verfahren nicht quadratisch konvergiert.

3.1. Fehleranalyse. Die Fehleranalyse für das Sekantenverfahren ist etwas aufwendiger als beim Newtonverfahren. Falls f zweimal differenzierbar und f'' stetig ist, kann man zeigen, daß die Iterationen

$$|x_{n+1} - \xi| \approx \frac{1}{2}\frac{|f''(\xi)|}{|f'(\xi)|}|x_n - \xi|^{1.62}$$

n	x_n	$f(x_n)$	$x_n - x_{n-1}$	$\frac{x_n - x_{n-1}}{x_{n-1} - x_{n-2}}$
0	0.00000000	$1.00e+000$		
1	1.57079633	$-7.92e-001$	$1.57e+000$	
2	0.87650155	$-3.52e-001$	$-6.94e-001$	$-2.81e-001$
3	0.32045471	$4.11e-001$	$-5.56e-001$	$-1.15e+000$
4	0.61981099	$-4.28e-002$	$2.99e-001$	$9.68e-001$
5	0.59154511	$-4.17e-003$	$-2.83e-002$	$-3.15e-001$
6	0.58849439	$5.32e-005$	$-3.05e-003$	$-3.82e+000$
7	0.58853279	$-6.42e-008$	$3.84e-005$	$4.13e+000$
8	0.58853274	$-9.86e-013$	$-4.63e-008$	$-3.14e+001$

TABELLE 3.1. Sekanten Verfahren für $f(x) = e^{-x} - \sin x$

erfüllen. Das Sekanten Verfahren konvergiert demnach langsamer als das Newtonverfahren, jedoch meist deutlich rascher als das Bisektionsverfahren. Bei der Wahl zwischen Newton– und Sekantenverfahren sollte aber auch der Aufwand für die Auswertung von f' berücksichtigt werden, da das Newton Verfahren zwar weniger Iterationen benötigt, jede Iteration aber mehr Zeit beansprucht als beim Sekantenverfahren.

4. Fixpunktiterationen

Ein **Fixpunkt** einer Abbildung $\mathfrak{I}\colon I \to \mathbb{K}$ ist eine Stelle $\xi \in I$ mit

$$\mathfrak{I}(\xi) = \xi,$$

ein Fixpunkt von $\mathfrak{I}$ wird also durch $\mathfrak{I}$ auf sich selbst abgebildet.

BEISPIEL 4.1. In der Abbildung 4.1 wird der Fixpunkt von $\mathfrak{I}(x) = \frac{1}{2} + x - \frac{x^2}{6}$, $0 \le x \le 3$, dargestellt als Durchschnitt der Graphen von $\mathfrak{I}$ und der sogenannten Identität, $id(x) = x$. Der Fixpunkt ist Lösung der Gleichung

$$x = \frac{1}{2} + x - \frac{x^2}{6},$$

welche äquivalent ist zu

$$x^2 = 3.$$

Der einzige Fixpunkt von $\mathfrak{I}$ ist daher $\xi = \sqrt{3}$. Ferner ist in Abb. 4.1 noch das Bild von $x_0 = 0.5$ unter $\mathfrak{I}$ eingezeichnet. Wegen $\mathfrak{I}(x_0) = 0.958\overline{3}$ liegt $x_1 = \mathfrak{I}(x_0)$ wieder im Definitionsbereich von $\mathfrak{I}$. Außerdem ist x_1 näher bei ξ als x_0. Man kann daher $x_2 = \mathfrak{I}(x_1) = \mathfrak{I}(\mathfrak{I}(x_0))$ bilden und erhält eine verbesserte Approximation von ξ, usw. Dies führt zu folgendem Algorithmus:

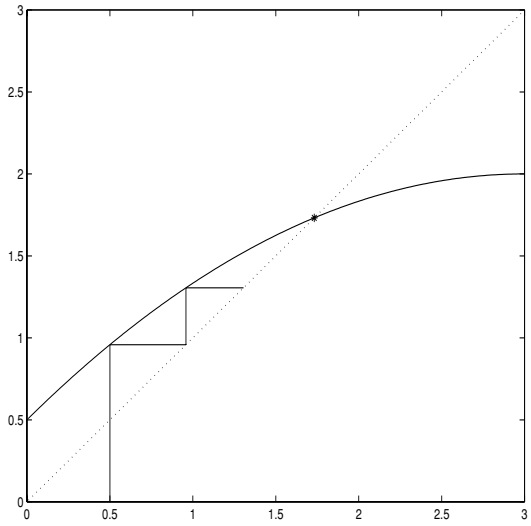


ABB. 4.1. Fixpunkt von $\frac{1}{2} + x - \frac{x^2}{6}$

ALGORITHMUS 4.1 (Sukkzessive Approximation). *Es sei* $\mathfrak{I}\colon I \to \mathbb{R}$ *eine stetige Funktion und* $tol > 0$ *eine gegebene Fehlertoleranz.*

Schritt 0: *Wähle Startwert* x_0

Schritt k: *Es steht* x_{k-1} *zur Verfügung*
Berechne x_k

$$x_k = \mathfrak{I}(x_{k-1}), \tag{4.1}$$

teste Abbruchbedingung

$$|x_{k-1} - x_k| \leq tol.$$

Angenommen, die Folge der Iterationen konvergiert, etwa gegen $\bar{x}$. Aus der Stetigkeit von $\mathfrak{I}$ folgt dann

$$\bar{x} = \lim_{k\to\infty} x_k = \lim_{k\to\infty} \mathfrak{I}(x_{k-1}) = \mathfrak{I}(\lim_{k\to\infty} x_{k-1}) = \mathfrak{I}(\bar{x}),$$

d.h. $\bar{x}$ ist ein Fixpunkt von $\mathfrak{I}$. Der Fixpunkt im vorigen Beispiel ist gleichzeitig Nullstelle der Gleichung

$$x^2 - 3 = 0.$$

Umgekehrt kann man diese Gleichung auf sehr verschiedene Weisen als Fixpunktgleichung $\mathfrak{I}_j(x) = x$, $j = 1, \ldots, 4$, schreiben:

(I1) $\mathfrak{I}_1 = \frac{1}{2} + x - \frac{x^2}{6}$
(I2) $\mathfrak{I}_2 = \frac{3}{x}$
(I3) $\mathfrak{I}_3 = \frac{2}{3}x^2 + x - 2$
(I4) $\mathfrak{I}_4 = \frac{1}{2}(x + \frac{3}{x})$.

n	$x_n^{(1)}$	$x_n^{(2)}$	$x_n^{(3)}$	$x_n^{(4)}$
0	2.0000	2.0000	2.0000	2.0000
1	1.8333	1.5000	2.6667	1.7500
2	1.7731	2.0000	5.4074	1.7321
3	1.7491	1.5000	22.9008	1.7321
4	1.7392	2.0000	370.5312	1.7321

TABELLE 4.1. Fixpunktiterationen

Man überzeuge sich davon, daß jede dieser Abbildungen nur $\xi = \sqrt{3}$ als Fixpunkt in $(0, 3)$ besitzt. Obwohl also diese Fixpunktgleichungen äquivalent sind zu $x^2 = 3$, ist das qualitative Verhalten der entsprechenden Fixpunktiterationen sehr unterschiedlich, vgl. Tabelle 4.1, in der $x_n^{(i)}$ die n–te Iterierte unter der Abbildung $\mathfrak{I}_i$, $i = 1, \ldots, 4$, bezeichnet.

Offensichtlich führen $\mathfrak{I}_1$ und $\mathfrak{I}_4$ zu konvergenten Iterationsfolgen (allerdings mit verschiedenen Konvergenzraten), $\mathfrak{I}_2$ erzeugt für diesen Startwert einen sogenannten Zyklus, während die Iterierten von $\mathfrak{I}_3$ divergieren. Die Stetigkeit von $\mathfrak{I}$ allein reicht also nicht aus, um die Konvergenz der Iterierten zu garantieren. Ein graphischer Vergleich klärt das unterschiedliche Verhalten dieser Iterationsvorschriften, Abb. 4.2: Der wesentliche Unterschied besteht im Anstieg von $\mathfrak{I}$ in der Nähe des Fixpunktes. $\mathfrak{I}_3$ ist sehr "steil", daher führen die Iterationen vom Fixpunkt weg, $\mathfrak{I}_1$ und $\mathfrak{I}_4$ verlaufen "flach", sodaß sich die jeweiligen Iterationen dem Fixpunkt nähern können. Die Annäherung erfolgt umso rascher, je "flacher" der Graph von $\mathfrak{I}$ in der Nähe des Fixpunktes ist. $\mathfrak{I}_2$ stellt einen seltenen Ausnahmefall dar. Diese Betrachtung führt zu der Vermutung, daß die Konvergenz der Iterationsfolge von der Größe des Anstiegs der Funktion in einer Umgebung des Fixpunktes beeinflußt wird. Der Wert der jeweiligen Abbildung ist

$$\mathfrak{I}_1'(\sqrt{3}) \approx 0.42, \qquad \mathfrak{I}_2'(\sqrt{3}) = -1,$$
$$\mathfrak{I}_3'(\sqrt{3}) \approx 3.31, \qquad \mathfrak{I}_4'(\sqrt{3}) = 0.$$

Eine weitere Voraussetzung, um das Verfahren überhaupt ansetzen zu können, wurde bereits erwähnt: Die Iterationen müssen natürlich wieder im Definitionsbereich von $\mathfrak{I}$ liegen. Dies ist sicherlich der Fall, wenn der Startwert x_0 in einem Intervall I liegt, das die Eigenschaft

$$\mathfrak{I}(I) \subset I$$

hat. Man sagt: I ist **invariant unter $\mathfrak{I}_4$**.

PROPOSITION 4.1. *Es sei $\mathfrak{I}\colon I \to \mathbb{R}$ stetig und $I = [a, b]$ invariant unter $\mathfrak{I}$. Dann hat $\mathfrak{I}$ mindestens einen Fixpunkt in I.*

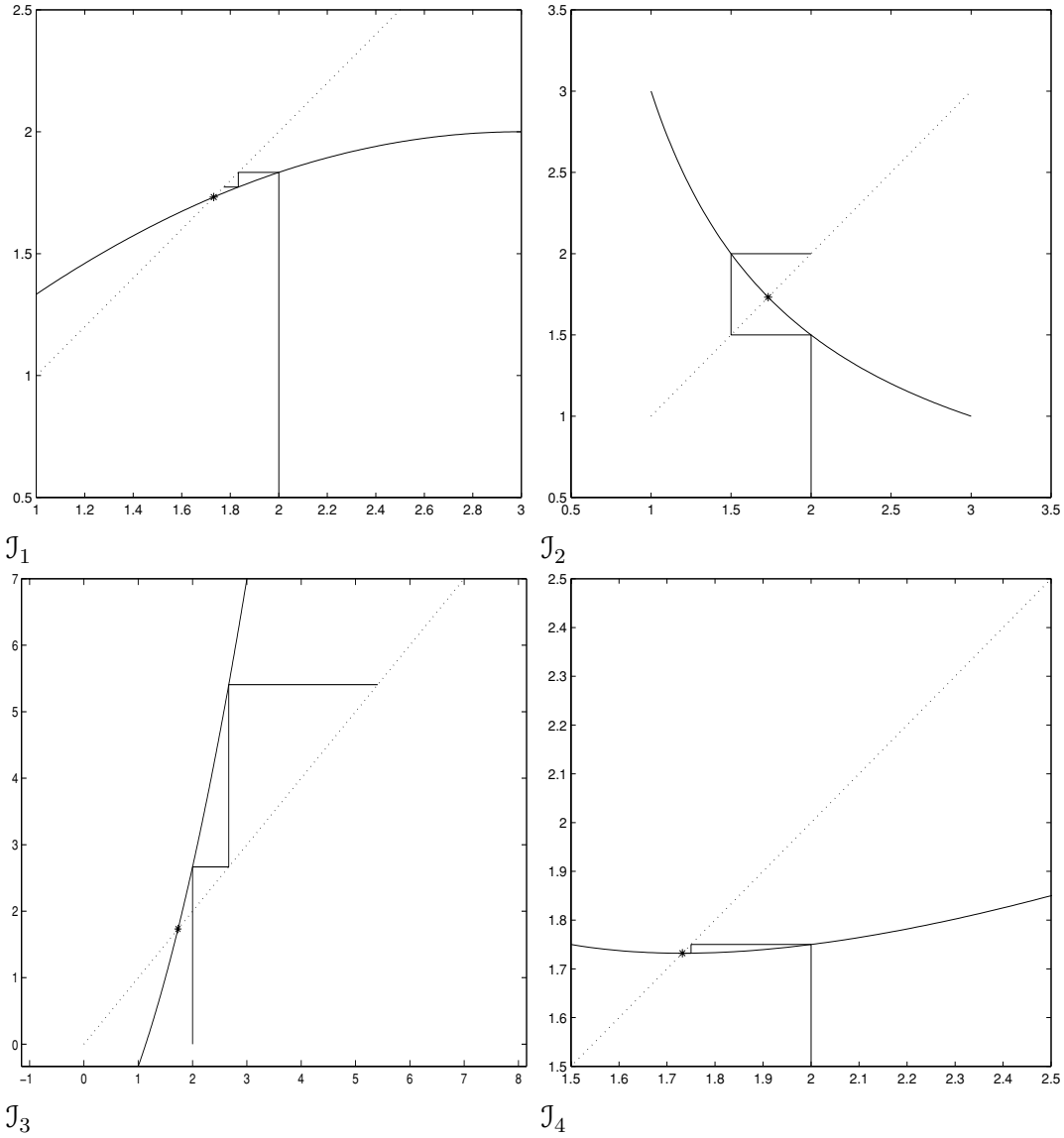


ABB. 4.2. Iterationen $\mathfrak{I}_1$ bis $\mathfrak{I}_4$

BEWEIS. Der Beweis ist eine einfache Anwendung des Zwischenwertsatzes 2.2 auf die Hilfsfunktion $g(x) = \mathfrak{I}(x) - x$. Man beachte: Eine Nullstelle von g ist ein Fixpunkt von $\mathfrak{I}$. Die Invarianz von I bedeutet, daß

$$a \leq \mathfrak{I}(x) \leq b$$

für alle $x \in [a,b]$ gilt, speziell also auch für $x = a$ und $x = b$. Dies hat

$$g(a) = \mathfrak{I}(a) - a \geq 0$$
$$g(b) = \mathfrak{I}(b) - b \leq 0$$

zur Folge. Da g stetig ist, sichert der Zwischenwertsatz die Existenz einer Nullstelle von g. □

Die Eindeutigkeit des Fixpunktes, als auch die Konvergenz der Iterationsfolge wird durch eine Schranke an die Ableitung von $\mathfrak{I}$ gesichert.

THEOREM 4.1 (Fixpunktsatz von Banach). *Es sei $\mathfrak{I}\colon I \to \mathbb{R}$ differenzierbar, f' stetig auf $I = [a,b]$ und I invariant unter $\mathfrak{I}$. Ferner gelte*

$$\lambda := \max_{x \in I} |\mathfrak{I}'(x)| < 1. \tag{4.2}$$

Dann gilt:

1. *$\mathfrak{I}$ besitzt in I genau einen Fixpunkt.*
2. *Die Iterationsfolge $x_n = \mathfrak{I}(x_{n-1})$, $n \geq 1$, konvergiert für jede Wahl des Startwertes $x_0 \in I$.*
3. *Der Fehler der n-ten Iteration genügt der Abschätzung*

$$|x_n - \xi| \leq \frac{\lambda^n}{1-\lambda} |x_0 - x_1| \tag{4.3}$$

4. *Die Iterationen konvergieren zumindest linear,*

$$\lim_{n \to \infty} \frac{\xi - x_{n+1}}{\xi - x_n} = \mathfrak{I}'(\xi) \tag{4.4}$$

BEWEIS. Der Beweis dieses Satzes ist etwas umfangreicher und soll hier nicht zur Gänze dargestellt werden. Wir begnügen uns darzulegen, wie die Zusatzbedingung (4.2) die Eindeutigkeit des Fixpunktes erzwingt. Dazu überlegen wir uns eine überaus nützliche Konsequenz aus (4.2): Es seien x, $y \in I$ beliebig gewählt, dann folgt mit dem Mittelwertsatz 3.2

$$\mathfrak{I}(x) - \mathfrak{I}(y) = \mathfrak{I}'(\tau)(x - y)$$

für ein $\tau \in I(x,y)$. Wegen der Stetigkeit von $\mathfrak{I}'$ und dem Satz von Weierstraß 2.3 ergibt sich die Ungleichung

$$|\mathfrak{I}(x) - \mathfrak{I}(y)| \leq \lambda |x - y|, \qquad x, y \in I. \tag{4.5}$$

Man sagt auch, $\mathfrak{I}$ erfüllt auf I eine **Lipschitz Bedingung**. Hat man eine Lipschitz Bedingung zur Verfügung, kann man einen Eindeutigkeitsbeweis folgendermaßen führen: Angenommen $\mathfrak{I}$ hätte zwei Fixpunkte, etwa ξ und η, (man beachte: Es wird nicht verlangt, daß $\mathfrak{I}$ tatsächlich einen Fixpunkt besitzt!), dann müßte für die Differenz gelten:

$$|\xi - \eta| = |\mathfrak{I}(\xi) - \mathfrak{I}(\eta)| \leq \lambda |\xi - \eta|.$$

Die Gleichheit ergibt sich aus der angenommenen Fixpunkteigenschaft von ξ und η, die Ungleichung folgt mit der Lipschitz Bedingung. Faktorisiert man $|\xi - \eta|$, erhält man

$$(1 - \lambda)|\xi - \eta| \leq 0. \tag{4.6}$$

Erst jetzt verwenden wir die Bedingung (4.2), $\lambda \in [0, 1)$: Diese hat nämlich $1 - \lambda > 0$ zur Folge. Da die linke Seite in (4.6) nicht negativ sein kann, muß daher zwangsläufig

$$|\xi - \eta| = 0$$

sein. Also gilt

$$\xi = \eta,$$

d.h. es kann höchstens einen Fixpunkt geben. □

Es ist meist recht schwierig, ein invariantes Intervall zu bestimmen, weshalb man in der Praxis Theorem 4.1 oft auf einem Intervall I anwendet, auf dem zumindest

$$\max_{x \in I} |\mathcal{J}'(x)| < 1$$

nachgewiesen werden kann und welches den gewünschten Fixpunkt enthält. In diesem Fall ist die Konvergenz der Iterationen nur mehr für Startwerte gesichert, welche hinreichend nahe bei ξ liegen. Für die konkrete Anwendung bedeutet dies, daß man versucht, einen möglichst guten Startwert zu finden und das Verfahren einfach ausprobiert.

Theorem 4.1 zeigt, daß die Fixpunktiterationen zumindest linear konvergieren, für $\mathcal{J}'(\xi) \neq 0$ ist allerdings nur lineare Konvergenz möglich. Aus diesem Grunde wird Fixpunktiteration nicht primär zur Nullstellenberechnung eingesetzt, für die wir bereits effizientere Verfahren kennengelernt haben. Abschließend sei noch bemerkt, daß man durch

$$\mathcal{J}(x) = x - \frac{f(x)}{f'(x)}.$$

auch das Newton Verfahren als Fixpunktiteration auffassen kann. Die Fixpunktiterationen stimmen dann mit den Newton Iterationen überein. Die Iteration $\mathcal{J}_4$ beispielsweise wurde durch Anwendung des Newtonverfahrens auf die Gleichung $x^2 - 3 = 0$ konstruiert. Berechnet man die Ableitung von $\mathcal{J}$ im Fixpunkt, erhält man

$$\mathcal{J}'(\xi) = 1 - \frac{(f')^2(\xi) - f(\xi)f''(\xi)}{(f')^2(\xi)} = 0,$$

(beachte $f(\xi) = 0$). Nach dem Fixpunktsatz von Banach konvergiert das Newton Verfahren daher mit höherer als linearer Ordnung.

n	x_n	τ_n	$\sqrt{3}-x_n$	Aitken
0	2.0000			
1	1.8333			
2	1.7731	0.3611	−0.0411	−0.0340
3	1.7491	0.3989	−0.0171	−0.0159
4	1.7392	0.4130	−0.0072	−0.0070
5	1.7351	0.4186	−0.0030	−0.0030
6	1.7333	0.4210	−0.0013	−0.0013
7	1.7326	0.4219	−0.0005	−0.0005

TABELLE 4.2. Fehler nach Aitken

4.1. Fehlerabschätzung und Konvergenzbeschleunigung nach Aitken. Wir haben bereits festgestellt, daß für die Fixpunktiterationen im Falle $|f'(\xi)| < 1$

$$\lim_{n\to\infty} \frac{\xi - x_n}{\xi - x_{n-1}} = \mathcal{J}'(\xi) \equiv \tau$$

gilt. Für hinreichend große n gilt daher

$$\frac{\xi - x_n}{\xi - x_{n-1}} \approx \tau.$$

Liest man dies als Gleichung, die nach ξ aufgelöst wird, ergibt sich

$$\xi \approx x_n + \frac{\tau}{1-\tau}(x_n - x_{n-1}). \tag{4.7}$$

Das unbekannnte τ kann für hinreichend große n aus

$$\tau_n = \frac{x_n - x_{n-1}}{x_{n-1} - x_{n-2}}$$

geschätzt werden. Wegen des Mittelwertsatzes folgt nämlich

$$\frac{x_n - x_{n-1}}{x_{n-1} - x_{n-2}} = \frac{\mathcal{J}(x_{n-1}) - \mathcal{J}(x_{n-2})}{x_{n-1} - x_{n-2}} = \mathcal{J}'(c_n)$$

mit einem $c_n \in I(x_{n-1}, x_{n-2})$. Wegen der Stetigkeit von $\mathcal{J}'$ konvergiert τ_n gegen τ und man kann τ in (4.7) durch τ_n ersetzen. Dies führt zu der Fehlerabschätzung nach Aitken

$$\xi - x_n \approx \frac{\tau_n}{1-\tau_n}(x_n - x_{n-1}), \quad n \geq 2. \tag{4.8}$$

In Tabelle 4.2 betrachten wir noch einmal die Iterierten unter $\mathcal{J}_1$. Als Abbruchbedingung wurde $|x_n - x_{n-1}| < 10^{-3}$ verwendet. Die Quotienten τ_n werden bereits ab der 4. Iteration stationär, daher stimmt auch die Aitkensche Fehlerabschätzung

exzellent mit dem tatsächlichen Fehler überein. Unsere theoretischen Überlegungen werden noch unterstützt durch

$$\mathcal{J}'(\sqrt{3}) \approx 0.4226.$$

Man kann die Fehlerabschätzung (4.8) aber auch verwenden, um die Iteration x_n zu korrigieren: Wäre die Darstellung des Fehlers (4.8) exakt, könnte man den Fixpunkt mit

$$\xi = x_n + (\xi - x_n)$$

berechnen. Wegen der Unschärfe in der Darstellung erhält man auf diese Weise jedoch nicht den exakten Fixpunkt, sondern eine Approximation $\hat{x}_n$, welche meist genauer ist als als x_n:

$$\hat{x}_n \approx x_n + \frac{\tau_n}{1-\tau_n}(x_n - x_{n-1}), \qquad \text{(Aitken Extrapolation)}. \tag{4.9}$$

Extrapoliert man beispielsweise x_4 aus Tabelle 4.2, erhält man (wegen der Übersichtlichkeit der Darstellung werden nur gerundete Zahlen angeschrieben, die tatsächliche Rechnung wurde mit der internen Genauigkeit von MATLAB durchgeführt)

$$\hat{x}_4 = 0.4130 + \frac{-0.0070}{1+0.0070}(-0.100) = 1.73259,$$

welches mit einem Fehler

$$\sqrt{3} - \hat{x}_4 = -1.99 \cdot 10^{-4}$$

behaftet ist. Dies ist deutlich genauer als jede weitere Iteration in Tabelle 4.2.

KAPITEL 3

Lineare Gleichungen

Systeme linearer Gleichungen sind nicht nur innerhalb der Mathematik ein fundamentaler Baustein, das Spektrum der Anwendungen umfaßt die Naturwissenschaften, Wirtschaft, Medizin usw. Es überrascht daher auch nicht, daß die Theorie linearer Gleichungen eine der am besten entwickelten mathematischen Disziplinen ist.

BEISPIEL 0.2. Ein einfaches Beispiel soll das Standardverfahren zur Lösung eines linearen Gleichungssystems in Erinnerung rufen:

$$\begin{array}{rcrcrcrcr} x_1 & + & x_2 & & & + & 3x_4 & = & 4 \\ 2x_1 & + & x_2 & - & x_3 & + & x_4 & = & 1 \\ 3x_1 & - & x_2 & - & x_3 & + & 2x_4 & = & -3 \\ -x_1 & + & 2x_2 & + & 3x_3 & - & x_4 & = & 4 \end{array} \tag{0.10}$$

Man benutzt die erste Gleichung, um die Unbekannte x_1 aus der 2. - 4. Gleichung zu eliminieren. Diese Wahl ist vorteilhaft, da der Koeffizient von x_1 in der ersten Gleichung 1 ist. Dazu multipliziert man der Reihe nach die erste Gleichung mit -2, -3 und 1 und addiert sie zur 2., 3. und 4. Gleichung. Dies ergibt das System

$$\begin{array}{rcrcrcrcr} x_1 & + & x_2 & & & + & 3x_4 & = & 4 \\ & - & x_2 & - & x_3 & - & 5x_4 & = & -7 \\ & - & 4x_2 & - & x_3 & - & 7x_4 & = & -15 \\ & + & 3x_2 & + & 3x_3 & + & 2x_4 & = & 8 \end{array}$$

Im nächsten Schritt eliminiert man die Unbekannte x_2 aus den Gleichungen 3 und 4, indem man die 2. Gleichung erst mit -4 und dann mit 3 multipliziert und zur 3. bzw. 4. Gleichung addiert. Das resultierende System lautet

$$\begin{array}{rcrcrcrcr} x_1 & + & x_2 & & & + & 3x_4 & = & 4 \\ & - & x_2 & - & x_3 & - & 5x_4 & = & -7 \\ & & & + & 3x_3 & + & 13x_4 & = & 13 \\ & & & & & - & 13x_4 & = & -13 \end{array}$$

Da nicht nur x_2 sondern (zufällig) auch x_3 aus der 4. Gleichung eliminiert wurde, kann das nun vorliegende System beginnend mit der letzten Gleichung bequem gelöst

werden. Man erhält der Reihe nach

$$\begin{aligned} x_4 &= 1 \\ x_3 &= \tfrac{1}{3}(13 - 13x_4) = \tfrac{1}{3}(13 - 13) = 0 \\ x_2 &= -(-7 + 5x_4 + x_3) = -(-7 + 5 + 0) = 2 \\ x_1 &= 4 - 3x_4 - x_2 = -1. \end{aligned}$$

Das Lösen eines linearen Gleichungssystems mit diesem Verfahren erfolgt also in zwei Phasen. In der *Eliminationsphase* versucht man durch sukkzessives Eliminieren von Unbekannten das gegebene System in ein äquivalentes **Dreieckssystem** überzuführen. Das anschließende Lösen des Dreieckssystems nennt man **Rücksubstitution**. Lineare Gleichungssysteme heißen **äquivalent**, wenn sie dieselbe Lösungsmenge besitzen. Die Lösungsmenge ändert sich nicht, wenn man

- die Reihenfolge zweier Gleichungen vertauscht,
- eine Gleichung mit einer von Null verschieden Zahl multipliziert,
- zu einer Gleichung ein Vielfaches einer anderen Gleichung addiert.

In den Rechnungen mit Papier und Bleistift wendet man diese Regeln in der Eliminationsphase so an, daß die resultierenden arithmetischen Operationen möglichst einfach sind. Man verwendet für die Elimination daher vorzugsweise Gleichungen, bei denen der Koeffizient der zu eliminierenden Unbekannten ± 1 ist, oder auf einfache Brüche führt. Derartige Bedenken spielen auf einem PC natürlich keine Rolle, im Vordergrund steht zunächst vielmehr, den Eliminationsprozeß möglichst systematisch zu gestalten. Doch zuvor muß man sich noch überlegen, wie man ein lineares Gleichungssystem am Rechner realisieren kann.

In den vorstehenden Gleichungssystemen wurde penibel darauf geachtet, die jeweiligen Unbekannten untereinander zu schreiben bzw. eine Lücke zu lassen, wenn eine Unbekannte in einer Gleichung nicht auftrat (gleichwertig hätte man diese Unbekannten auch mit einem Koeffizienten 0 berücksichtigen können). Man kann nun ohne Informationsverlust auf das Anschreiben der Unbekannten verzichten, wenn man vereinbart, die erweiterte Koeffizientenmatrix

$$\begin{array}{rrrrr} 1 & 1 & 0 & 3 & 4 \\ 2 & 1 & -1 & 1 & 1 \\ 3 & -1 & -1 & 2 & -3 \\ -1 & 2 & 3 & -1 & 4 \end{array}$$

als lineares Gleichungssystem (0.10) zu interpretieren. Man überzeugt sich leicht davon, daß den Manipulationen von Gleichungen in der Eliminationsphase dieselben Manipulationen der Zeilen der erweiterten Koeffizientenmatrix entsprechen. In dieser Form kann man den beschriebenen Lösungsweg zu einem allgemeinen Lösungsalgorithmus für Systeme von n Gleichungen in m Unbekannten ausbauen.

1. Das Gaußsche Eliminationsverfahren

Wir betrachten das System von n Gleichungen in n Unbekannten x_1, ..., x_n

$$\begin{array}{rcl} a_{11}x_1 + \cdots + a_{1n}x_n & = & b_1 \\ \vdots & \vdots & \vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n & = & b_n \end{array} \tag{1.1}$$

mit $a_{ij}, b_j \in \mathbb{K}$, $i = 1, \ldots, n$, $j = 1, \ldots, n$. Wir fassen die Koeffizienten zu einer $n \times n$ Matrix A zusammen, die rechten Seiten der Gleichungen zu einem Vektor b. Die Schreibweise $A \in \mathbb{K}^{n\times m}$ bedeutet, daß A eine Matrix mit n Zeilen und m Spalten ist und die Matrixelemente aus $\mathbb{K}$ genommen werden. Mit $b \in \mathbb{K}^n$ deuten wir an, daß b ein n-Tupel von Zahlen aus $\mathbb{K}$ ist. Man nennt A **Koeffizientenmatrix** von (1.1), $b \neq 0$ **Inhomogenität** von (1.1) und $[A\,b]$ **erweiterte Koeffizientenmatrix** (an die Matrix A wird b als $m+1$-te Spalte angehängt). In der erweiterten Koeffizientenmatrix ist die gesamte Information des linearen Gleichungssystems (1.1) kodiert. In der linearen Algebra wird gezeigt, wie man aus der erweiterten Koeffizientenmatrix Eigenschaften über mögliche Lösungen von (1.1) wie Existenz, Eindeutigkeit, etc. ablesen kann. Darauf soll hier nicht eingegangen werden. Ziel des Eliminationsverfahrens ist, das Gleichungssystem (1.1) schrittweise in ein äquivalentes **Dreieckssystem**, also ein System der speziellen Struktur

$$\begin{array}{rcccccl} \tilde{a}_{11}x_1 & +\tilde{a}_{12}x_2 & + & \cdots & +\tilde{a}_{1n}x_n & = \tilde{b}_1 \\ & \tilde{a}_{22}x_2 & + & \cdots & +\tilde{a}_{2n}x_n & = \tilde{b}_2 \\ & & & \ddots & \vdots & \vdots \\ & & & & \tilde{a}_{nn}x_n & = \tilde{b}_n \end{array} \tag{1.2}$$

umzuwandeln. Dieses kann mit **Rücksubstitution** effizient gelöst werden, soferne die *Diagonalelemente* $\tilde{a}_{ii}$ der Koeffizientenmatrix nicht Null sind:

Algorithmus 1.1 (Rücksubstitution). *Löse* (1.2) *schrittweise nach* x_n, x_{n-1}, ... ,x_1 *(der Übersichtlichkeit halber bezeichnen wir die Elemente der erweiterten Matrix wieder mit* a_{ij}, b_i*):*

$$x_n = \frac{b_n}{a_{nn}} \tag{1.3}$$

$$x_i = \frac{1}{a_{ii}}(b_i - \sum_{k=i+1}^{n} a_{ik}x_k). \tag{1.4}$$

Bezeichnet man mit $[A^{(k)}\,b^{(k)}]$ die resultierende erweiterte Koeffizientenmatrix zu Beginn des k-ten Eliminationsschrittes und setzt man

$$[A^{(1)}\,b^{(1)}] = [A\,b],$$

kann man das Gaußsche Eliminationsverfahren folgendermaßen beschreiben:

ALGORITHMUS 1.2 (Eliminationsverfahren nach Gauß). *Zu Beginn des k–ten Schrittes, k=1, ... ,n-1, liege das Gleichungssystem in folgender Form vor:*

$$[A^{(k)}\, b^{(k)}] = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ & a_{22}^{(2)} & & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ & & \ddots & & \vdots & \vdots \\ & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} & b_k^{(k)} \\ & & \vdots & & \vdots & \vdots \\ & & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} & b_n^{(k)} \end{pmatrix}.$$

Ist $a_{kk}^{(k)} \neq 0$, eliminiert man x_k aus den Gleichungen $k+1$, ... ,n mittels:

$$\begin{aligned} m_{ik} &= \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, && i = k+1, \ldots, n \\ a_{ij}^{(k+1)} &= a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}, && j = k+1, \ldots, n \\ b_i^{(k+1)} &= b_i^{(k)} - m_{ik} b_k^{(k)} \end{aligned} \tag{1.5}$$

(Matrixelemente in $[A^{(k)}\, b^{(k)}]$, welche auf Grund des Algorithmus Null sind, wurden nicht angeschrieben). Die Diagonalelemente $a_{kk}^{(k)}$ nennt man **Pivotelemente**.

Man kann sich überlegen, daß der Rechenaufwand für die Eliminationsphase in der Größenordnung von $\frac{2}{3}n^3$ arithmetischen Operationen liegt, die Rücksubstitution benötigt dagegen nur etwa n^2 Operationen.

In dieser Form ist der Algorithmus jedoch noch nicht brauchbar, da die Voraussetzung $a_{kk}^{(k)} \neq 0$ selbst bei so einfachen Beispielen wie

$$\begin{aligned} y &= 1 \\ x + y &= 2 \end{aligned}$$

einen Abbruch erzwingt. Das Problem wird behoben durch Vertauschen der beiden Gleichungen.

1.1. Spaltenpivotsuche. Eine Vertauschung der Reihenfolge von zwei Gleichungen kann aber auch aus numerischen Gründen angebracht sein. Dies liegt daran, daß auf einem Computer stets nur eine endliche Stellenzahl zur Verfügung steht. Wir demonstrieren den Effekt der endlichen Computerarithmetik an einem einfachen Beispiel. Wir nehmen an, daß sämtliche Rechnungen auf einem dezimalen Computer durchgeführt werden, welcher 4 Stellen der Mantisse speichern kann. Maschinenintern werde also jede Zahl in der Form

$$\pm 0.d_1 d_2 d_3 d_4 \cdot 10^e, \quad d_1 \neq 0,$$

mit $d_i \in \{0, 1, \ldots, 9\}$, $i = 1, \ldots, 4$, dargestellt. Zahlen,welche nicht diesem Format entsprechen, müssen daher auf 4 Stellen gerundet werden, ehe sie weiter verarbeitet

werden können. Dabei entstehen **Rundungsfehler**, die das Ergebnis einer Rechnung erheblich verfälschen können. Beispielsweise ergibt $1000 + 0.6$ nicht 1000.6 sondern 1001. Welche Konsequenzen ergeben sich daraus für den Gaußschen Algorithmus?

Beispiel 1.1. Wir betrachten das Gleichungssystem

$$\begin{array}{rrcr} 0.003000x_1 & +59.14x_2 & = & 59.17 \\ 5.291x_1 & -6.130x_2 & = & 46.78, \end{array}$$

das die exakte Lösung $x_1 = 10$ und $x_2 = 1$ besitzt. Zur Elimination von x_1 aus der 2. Gleichung wäre die erste Gleichung mit $m_{21} = \frac{5.291}{0.003000} = 1763.6\dot{6}$ zu multiplizieren. Tatsächlich rechnet man mit $m_{21} = 1764$. Das berechnete Dreiecksystem hat demnach die erweiterte Koeffizientenmatrix

$$\begin{array}{rrr} 0.003000 & 59.14 & 59.17 \\ & -104300 & -104400 \end{array}$$

$$\begin{aligned} &(-6.130 - 59.14 \cdot 1764 = -6.130 - 104322.96 \doteq -6.130 - 104300 \doteq 104300, \\ &46.78 - 59.17 \cdot 1764 = 46.78 - 104375.88 \doteq 46.78 - 104400 \doteq 104400), \end{aligned}$$

während die exakte Rechnung

$$\begin{array}{rrr} 0.003000 & 59.14 & 59.17 \\ & -104309.37\dot{6} & -104309.37\dot{6} \end{array}$$

liefert. Rücksubstitution ergibt

$$\begin{aligned} &\hat{x}_2 \doteq 1.001 \\ &\hat{x}_1 = \frac{59.17 - 59.14 \cdot 1.001}{0.003000} \doteq \frac{59.17 - 59.20}{0.003000} \doteq -10.00(!), \end{aligned} \tag{1.6}$$

$\hat{x}_1$ stimmt in keiner einzigen signifikanten Stelle mit x_1 überein, nicht einmal das Vorzeichen ist korrekt! Analysiert man diese Rechnung etwas genauer, erkennt man, daß die Ursache für den dramatischen Fehlschlag die unterschiedliche Größenordnung des Pivotelementes a_{11} im Vergleich zu den anderen relevanten Matrixelementen (hier nur a_{22}, b_2) ist. Dadurch wird der Multiplikator m_{21} so groß, daß bei der Berechnung des Dreieckssystems a_{22} und b_2 der Rundung zum Opfer fallen und nicht verwendet werden können. Aber auch bei der Rücksubstitution hat die Kleinheit des Pivotelementes verheerende Auswirkungen: Schreibt man (1.6) in der Form

$$\hat{x}_1 = \frac{59.17 - 59.14(x_2 + 0.001)}{0.003000} \doteq x_1 - \frac{59.14}{0.003000} 0.001,$$

zeigt sich, daß der an sich kleine Fehler in x_2 bei der Berechnung von $\hat{x}_1$ um den Faktor $\frac{59.14}{0.003000} \approx 19700$ verstärkt wird.

Der Einwand, der Einfluß von Rundungsfehlern sei maßlos überzeichnet, da moderne Computer nicht mit 4 sondern ca. 16 Dezimalstellen rechnen, ist nicht gerechtfertigt. Zum einen läßt sich für jede Rechengenauigkeit ein Gleichungssystem angeben, bei dem der demonstrierte Effekt auftritt. Zum anderen wird für große Systeme

($n \geq 100$) die Komplexität des Algorithmus spürbar: Es gibt etwa $\frac{2}{3}n^3$ möglicherweise mit Rundungsfehlern behaftete Rechenoperationen. Um den Einfluß dieser Rundungsfehler zu minimieren wurden verschiedene Strategien zur Wahl des Pivotelementes entwickelt. Die Diskussion des Beispiels legt folgende Vorgangsweise nahe:

ALGORITHMUS 1.3 (Teilpivotsuche). *Modifiziere im k-ten Schritt* $[A^{(k)}\, b^{(k)}]$ *folgendermaßen:*

	Bestimme den kleinsten Index j_0, $k \leq j_0 \leq n$, *mit*
Schritt k_1	$\lvert a_{j_0 k}\rvert = \max\limits_{k \leq j \leq n} \lvert a_{jk}\rvert$
Schritt k_2	*Vertausche die Zeilen* j_0 *und* k *von* $[A^{(k)}\, b^{(k)}]$
Schritt k_3	*Gaußelimination* (1.5)

Im Fall $j_0 = k$ ist natürlich keine Vertauschung erforderlich. Die Minimaleigenschaft von j_0 macht den Algorithmus eindeutig, wenn das Maximum in dem betreffenden Teil der k-ten Spalte mehrfach angenommen wird. Tritt der Fall

$$\max_{k \leq j \leq n} |a_{jk}| = 0$$

ein, kann kein zulässiges Pivotelement gefunden werden und das Gleichungssystem besitzt entweder keine oder unendlich viele Lösungen. In dieser Situation sollte das Programm mit einem entsprechenden Hinweis beendet werden.

Wendet man die Teilpivotsuche auf das Gleichungssystem aus Beispiel 1.1 an, d.h. vertauscht man die Reihenfolge der beiden Gleichungen, ergibt sich

$$\begin{array}{rrrcrrrcll} 5.291 & -6.130 & 46.78 & \rightarrow & 5.291 & -6.130 & 46.78 & \rightarrow & \hat{x}_1 & = 10.00 \\ 0.003000 & 59.14 & 59.17 & & & 59.14 & 59.14 & & \hat{x}_2 & = 1.000 \end{array}$$

Trotz vierstelliger Arithmetik erhalten wir das korrekte Resultat. Der Multiplikator ist nun $m_{21} = 0.5670 \cdot 10^{-3}$, bei der Elimination rechnet man

$$59.14 + 0.0005670 \cdot 6.130 \doteq 59.14 + 0.003476 \doteq 59.14$$
$$59.17 - 0.0005670 \cdot 46.78 \doteq 59.17 - 0.02652 \doteq 59.17 - 0.03 \doteq 59.14,$$

die Rücksubstitution ergibt $\hat{x}_2 = 1.000$ und

$$\hat{x}_1 = \frac{46.78 + 6.130}{5.291} = \frac{52.91}{5.291}.$$

In diesem Abschnitt haben wir uns mit dem Einfluß unvermeidlicher Rundungsfehler beschäftigt. Verfahren, bei welchen Rundungsfehler zu so fatalen Konsequenzen führen können, nennt man **(numerisch) instabil**. Verfahren, welche unempfindlich sind gegenüber Rundungsfehlern, nennt man **stabil**.

2. Grundlagen der Linearen Algebra

Bisher wurden Matrizen nur als bequeme Schreibweise für lineare Gleichungssysteme verwendet. Eine Änderung des Blickwinkels bringt nicht nur mehr Struktur,

sondern auch mehr Einsicht. Wir setzen eine gewisse Vertrautheit im Umgang mit Vektoren voraus.

DEFINITION 2.1. *Eine Abbildung* $f\colon \mathbb{K}^m \to \mathbb{K}^n$ *heiß* **linear**, *genau dann wenn*

- $f(\lambda x) = \lambda f(x)$
- $f(x+y) = f(x) + f(y)$

für alle $\lambda \in \mathbb{K}$ *und* $x, y \in \mathbb{K}^m$ *zutrifft.*

BEISPIEL 2.1. Es sei $A \in \mathbb{K}^{n\times m}$ eine Matrix. Die Abbildung $f_A\colon \mathbb{K}^m \to \mathbb{K}^n$ sei für $x = (\xi_1, \dots, \xi_m) \in \mathbb{K}^m$ definiert durch

$$(f_A(x))_i = \sum_{k=1}^{m} a_{ik}\xi_k, \quad i = 1, \dots, n \tag{2.1}$$

Man überzeugt sich leicht davon, daß f_A linear ist. Anstelle von $f_A(x)$ schreibt man üblicherweise Ax. Umgekehrt wird in der Linearen Algebra gezeigt, daß jede lineare Abbildung $f\colon \mathbb{K}^m \to \mathbb{K}^n$ durch eine Matrix $A \in \mathbb{K}^{n\times m}$ beschrieben werden kann.

Lineare Abbildungen können addiert und mit einem Skalar multipliziert werden. Dies überträgt sich sinngemäß auf Matrizen: für $A, B \in \mathbb{K}^{n\times m}$ und $\lambda \in \mathbb{K}$ folgt also

$$(\lambda A)_{ij} = \lambda A_{ij}$$
$$(A+B)_{ij} = A_{ij} + B_{ij}.$$

Interessanteres ergibt die Verknüpfung zweier linearer Abbildungen $f_A\colon \mathbb{K}^m \to \mathbb{K}^n$ und $f_B\colon \mathbb{K}^r \to \mathbb{K}^m$, welchen die Matrizen $A \in \mathbb{K}^{n\times m}$ und $B \in \mathbb{K}^{m\times r}$ entsprechen. Wir bezeichnen die Element von A bzw. B mit a_{ij} bzw. b_{ij}. Es sei h die Komposition $f_A \circ f_B\colon \mathbb{K}^r \to \mathbb{K}^n$. Dann ist h linear und für $x = (\xi_1, \dots, \xi_r) \in \mathbb{K}^r$ ergibt sich mit (2.1)

$$\begin{aligned} h(x)_i &= (f_A(f_B(x)))_i = \\ &= \sum_{k=1}^{m} a_{ik}(f_B(x))_k = \sum_{k=1}^{m} a_{ik} \sum_{j=1}^{r} b_{kj}\xi_j \\ &= \sum_{j=1}^{r} \big(\sum_{k=1}^{m} a_{ik}b_{kj}\big)\xi_j, \quad i = 1, \dots, n. \end{aligned}$$

Definiert man also die Produktmatrix $C \in \mathbb{K}^{n\times r}$, $C = AB$, durch

$$c_{ij} = \sum_{k=1}^{m} a_{ik}b_{kj} \tag{2.2}$$

erhält man

$$h(x)_i = \sum_{j=1}^{r} c_{ij}\xi_j, \quad i = 1, \dots, n.$$

Der Hintereinanderausführung der linearen Abbildungen f_A und f_B entspricht daher das Matrizenprodukt AB. Da natürlich im allgemeinen $f_A \circ f_B \neq f_B \circ f_A$ ist, kann auch $AB = BA$ nicht allgemein gelten. Das Matrizenprodukt ist nicht kommutativ.

BEMERKUNG 2.1. In MATLAB realisiert man das Produkt zweier Matrizen A und B durch die Anweisung

$$\mathrm{C} = \mathrm{A} * \mathrm{B};$$

Man achte auf den Unterschied in der Wirkung von A .* B und A * B !

Wir betrachten nun die lineare Abbildung $f_A \colon \mathbb{K}^n \to \mathbb{K}^n$ und die zugehörige Matrix $A \in \mathbb{K}^{n\times n}$. Man kann zeigen, daß f_A genau dann injektiv ist, wenn f_A surjektiv ist. Ist f_A injektiv, existiert die Umkehrabbildung f_A^{-1} und ist auf ganz $\mathbb{K}^n$ definiert. Da die Umkehrabbildung einer linearen Abbildung wieder linear ist, entspricht der Abbildung f_A^{-1} eine Matrix $A^{-1} \in \mathbb{K}^{n\times n}$. Diese ist eindeutig bestimmt und man nennt A^{-1} die zu A **inverse Matrix**. Die charakterisierende Eigenschaft der Umkehrabbildung

$$f_A \circ f_A^{-1} = f_A^{-1} \circ f_A = id$$

charakterisiert auch die inverse Matrix

$$AA^{-1} = A^{-1}A = E.$$

Dabei steht E für die *Einheitsmatrix*, i.e. $a_{ii} = 1, a_{ij} = 0$ für $i \neq j$. Man kann sogar zeigen, daß eine Matrix B, welche nur eine der beiden Gleichungen

$$AB = E, \qquad BA = E$$

erfüllt, die inverse Matrix A^{-1} ist, und daher automatisch auch die zweite Gleichung erfüllt. Eine Matrix $A \in \mathbb{K}^{n\times n}$ heißt **regulär**, wenn sie invertierbar ist. Andernfalls nennt man A **singulär**. Ein lineares Gleichungssystem wie z.B. (1.1) läßt sich nun kompakt schreiben in der Form

$$Ax = b.$$

3. LU Faktorisierung

Wir wenden den Gaußschen Algorithmus auf das Gleichungssystem

$$Ax = b,$$

$A \in \mathbb{K}^{n\times n}$, an, und setzen voraus, daß die Reduktion zu einem Dreiecksystem ohne Zeilenvertauschungen durchführbar ist. Die Matrix $U \in \mathbb{K}^{n\times n}$ (U steht für upper triangular) sei die resultierende Koeffizientenmatrix

$$U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn}, \end{pmatrix}$$

mit $u_{kk} = a_{kk}^{(k)}$. Ferner sammeln wir die benötigten Multiplikatoren in einer unteren Dreiecksmatrix $L \in \mathbb{K}^{n\times n}$ (lower triangular):

$$L = \begin{pmatrix} 1 & & & \\ m_{21} & 1 & & \\ \vdots & \ddots & & \\ m_{n1} & \dots & m_{n,n-1} & 1 \end{pmatrix}$$

Zwischen den Matrizen A, L, U besteht die Beziehung

THEOREM 3.1 (LU Faktorisierung). *Es sei A regulär und der Gaußsche Algorithmus sei ohne Zeilenvertauschungen ausführbar. Dann gilt*

$$A = LU.$$

Eine Matrix A heißt **strikt diagonaldominant**, wenn

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| \tag{3.1}$$

für alle $i = 1, \dots, n$ gilt. Für strikt diagonaldominante Matrizen ist der Gaußsche Algorithmus ohne Zeilenvertauschungen ausführbar.

Hat man die LU Faktorisierung berechnet, folgt für jedes $x \in \mathbb{K}^n$ $Ax = L(Ux)$. Setzt man $Ux = y$, ist $Ax = b$ gleichwertig mit den zwei Dreieckssystemem

$$Ly = b \tag{3.2}$$

$$Ux = y. \tag{3.3}$$

Das System $Ly = b$ löst man schrittweise, beginnend mit der ersten Gleichung (**Vorwärtseinsetzen**).

3.1. Mehrere Inhomogenitäten. Das Abspeichern der LU Faktorisierung ist besonders dann vorteilhaft, wenn beabsichtigt ist, dasselbe Gleichungssystem mit verschiedenen Inhomogenitäten $b_1, \dots, b_r$ zu lösen. In diesem Falle hat man die rechenintensive LU Faktorisierung nur einmal zu durchzuführen und löst dann die Systeme

$$\begin{aligned} Ly_i &= b_i \\ Ux_i &= y_i, \end{aligned} \tag{3.4}$$

$i = 1, \dots, r$. In MATLAB können die Systeme (3.4) simultan gelöst werden, indem man die Inhomogenitäten b_i in einer Matrix $B = [b_1 \cdots b_r]$ zusammenfaßt und die Vorwärts– bzw. Rückwärtssubstitution auf die *Matrizengleichungen*

$$\begin{aligned} LY &= B \\ UX &= Y \end{aligned}$$

anwendet. Die Spalten von X sind dann die Lösungen x_i der Systeme (3.4). Werden die Faktoren L und U nicht benötigt, kann man den Gauß'schen Algorithmus direkt auf das System anwenden, welches B als Inhomogenität besitzt.

3.2. Inverse Matrix. Die inverse Matrix spielt zwar in der Theorie der linearen Gleichungen eine prominente Rolle, denn die Lösung von $Ax = b$ ist, falls A invertierbar ist, explizit darstellbar durch

$$x = A^{-1}b. \tag{3.5}$$

Trotzdem ist es in der Praxis nur sehr selten notwendig, A^{-1} explizit zu berechnen. Ist etwa der Vektor z bestimmt durch

$$z = A^{-1}y$$

für ein gegebenes $y \in \mathbb{K}^n$, berechnet man nicht zuerst A^{-1} und multipliziert dann A^{-1} mit y, sondern man löst staatt dessen das Gleichungssytem

$$Az = y.$$

Die Berechnung von A^{-1} erfordert nämlich selbst das Lösen von n simultanen linearen Gleichungssystemen. Um dies einzusehen, erinnern wir uns daran, daß A^{-1} die Lösung der Matrizengleichung

$$AX = E \tag{3.6}$$

ist. Bezeichnet man mit x_i bzw. e_i die Spalten von X bzw. E, kann man die Matrixgleichung (3.6) als simultanes Gleichungssystem für die Spalten der inversen Matrix interpretieren:

$$Ax_i = e_i, \quad i = 1, \ldots, n$$

3.3. Die Determinante einer Matrix. Eine theoretisch bedeutsame Kenngröße einer Matrix ist ihre Determinante $\det(A)$. Beispielsweise ist eine Matrix A genau dann singulär, wenn $\det(A) = 0$ ist. Allerdings ist der Rechenaufwand für die Berechnung der Determinante einer $n \times n$ Matrix an Hand ihrer Definition proportional zu $n!$. Dies ist einer der Gründe, weshalb Determinanten in der Praxis nur geringe Bedeutung haben. Sollte ihre Berechnung tatsächlich einmal notwendig sein, kann man auf die LU-Zerlegung zurückgreifen. Steht die Faktorisierung

$$A = LU$$

zur Verfügung, folgt aus den Rechenregeln für Determinanten

$$\det(A) = u_{11} \cdot u_{22} \cdots u_{nn} \tag{3.7}$$

4. Die Konditionierung eines linearen Gleichungssystems

Wir wenden uns nun einem Phänomen zu, dessen Auswirkungen ähnlich katastrophal sein können wie die numerische Instabilität der naiven Gauß Elimination, welches aber gänzlich verschiedene Ursachen hat. Wir beginnen wieder mit einem einfachen Beispiel:

Beispiel 4.1. Wir betrachten das 2×2 System mit vorerst noch unbestimmter Inhomogenität $b = (\beta_1\ \beta_2)^T$:

$$\begin{aligned} x_1 - 11x_2 &= \beta_1 \\ -9x_1 + 100x_2 &= \beta_2 \end{aligned} \tag{4.1}$$

Die Inverse einer 2×2 Matrix $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ mit $\det A = ad - bc \neq 0$ ist gegeben durch

$$A^{-1} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}.$$

Die exakte Lösung von (4.1) ist daher (vgl. (3.5))

$$x = (x_1 x_2) = \frac{1}{100-99} \begin{pmatrix} 100 & 11 \\ 9 & 1 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 100\beta_1 + 11\beta_2 \\ 9\beta_1 + \beta_2 \end{pmatrix} \tag{4.2}$$

Für die Inhomogenitäten $b = \left(\begin{smallmatrix} 0 \\ 10 \end{smallmatrix}\right)$ und $d = \left(\begin{smallmatrix} -12 \\ 109 \end{smallmatrix}\right)$ ergibt dies die Lösungen

$$x = \begin{pmatrix} 110 \\ 10 \end{pmatrix}, \qquad y = \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

Wir stören nun die Inhomogenitäten und betrachten

$$\hat{b} = \begin{pmatrix} 0 \\ 10.1 \end{pmatrix}, \qquad \hat{d} = \begin{pmatrix} -13 \\ 109 \end{pmatrix}.$$

Dies führt auf die Lösungen

$$\hat{x} = \begin{pmatrix} 111.1 \\ 10.1 \end{pmatrix}, \qquad \hat{y} = \begin{pmatrix} -101 \\ -8 \end{pmatrix}.$$

Betrachten wir nun die relativen Änderungen in den Lösungen: Für x_1 findet man

$$\begin{aligned} \frac{\hat{x}_1 - x_1}{x_1} &= \frac{111.1 - 110}{110} = 0.01 \\ \frac{\hat{x}_2 - x_2}{x_2} &= \frac{10.1 - 10}{10} = 0.01. \end{aligned}$$

In diesem Falle ist also die relative Änderung der Lösung von derselben Größenordnung wie die verursachende Störung. Ganz anders sind die Auswirkungen der Störung um

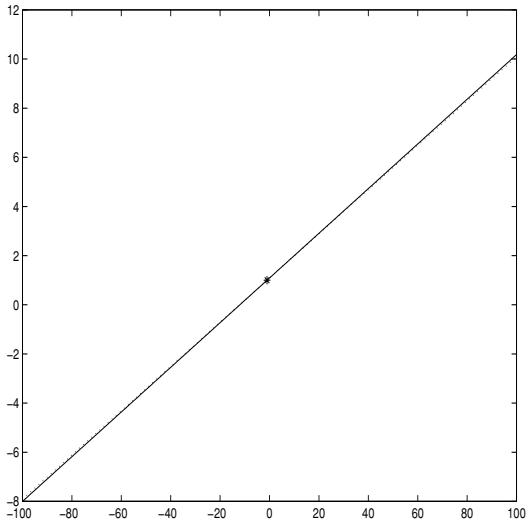


ABB. 4.1. Schlechte Kondition eines LGS

rund 10% von d:

$$\frac{\hat{y}_1 - y_1}{y_1} = \frac{-101 - (-1)}{-1} = 100,$$
$$\frac{\hat{y}_1 - y_1}{y_2} = \frac{-8 - 1}{1} = -9$$

eine Störung von d um rund 10% bewirkt Veränderung in den Lösungen im Ausmaß von 10 000% bzw. 900%! Die Wirkung der Störung übertrifft die Ursache etwa um einen Faktor 1000: Das Gleichungssystem (4.1) ist **schlecht konditioniert**, d.h. die Lösung ist sehr sensibel gegenüber manchen Störungen in den Daten (das sind hier Koeffizienten und Inhomogenität). In dieser einfachen Situation kann man die schlechte Kondition geometrisch veranschaulichen (Abb. 4.1). Die Gleichungen werden durch nahezu parallele Gerade dargestellt, deren schleifender Schnitt auf jede Lageänderung einer der Geraden überproportional reagiert.

Direkter Vergleich durch explizites Berechnen der Lösungen ist mit vernünftigen Aufwand leider auf 2×2 Systeme beschränkt. Wie kann man also auch für große Systeme feststellen, ob sie gut oder schlecht konditioniert sind? Kann man die Kondition einer Matrix quantifizieren und damit Matrizen in dieser Hinsicht vergleichbar machen? Zur Beantwortung dieser Fragen wurden verschiedene Konditionszahlen entwickelt.

4.1. Konditionszahlen. Aufbauend auf geometrische Anschauung läßt sich eine Konditionszahl angeben, welche für 2×2 Systeme ein Maß für den Winkel zwischen

den beiden Geraden reprsentiert. Die Kondition einer Matrix kann dann durch

3)
$$V = \frac{|\det(A)|}{\alpha_1 \alpha_2 \cdots \alpha_n}$$

wobei

$$\alpha_i = \sqrt{a_{i1}^2 + \cdots + a_{in}^2}, \quad i = 1, \ldots, n$$

gemessen werden. Aus der Herleitung ergibt sich, daß für dieses "Volumsmaß" stets

$$0 \leq V \leq 1$$

gilt. Eine Matrix ist umso schlechter konditioniert, je kleiner V ist. $V = 1$ entspricht einer perfekt konditionierten Matrix, z.B. der Einheitsmatrix, $V = 0$ einer singulären Matrix.

Beispiel 4.2. Für die Matrix zu (4.1) erhält man

$$\begin{aligned} \alpha_1 &= \sqrt{1 + 121} \doteq 11.05, \\ \alpha_2 &= \sqrt{10000 + 81} \doteq 100.40 \\ \det(A) &= 1 \end{aligned}$$

und somit

$$V \doteq 0.00090,$$

ein eindeutiges Warnsignal vor der schlechten Kondition von A.

Das "Volumsmaß" ist zwar relativ einfach zu berechnen, gibt aber keinen Hinweis über die Größenordnung der möglichen Auswirkungen einer schlechten Kondition. Um auch darüber Auskunft zu bekommen, führen wir folgende Analyse durch, für die wir den Begriff Norm eines Vektors und einer Matrix benötigen. Die euklidische Länge $\sqrt{\xi_1^2 + \cdots + \xi_n^2}$ eines Vektors $x = (\xi_1, \ldots, \xi_n)^T \in \mathbb{K}^n$ ist intuitiv durch einige Eigenschaften charakterisiert, die sich in Form der folgenden Axiome präzisieren lassen:

$$\begin{aligned} &\text{(N1)} \quad ||x|| \geq 0 \\ &\text{(N2)} \quad ||x|| = 0 \Rightarrow x = 0 \\ &\text{(N3)} \quad ||\lambda x|| = |\lambda| ||x||, \quad \lambda \in \mathbb{K} \\ &\text{(N4)} \quad ||x + y|| \leq ||x|| + ||y||, \quad y \in \mathbb{K}^n. \end{aligned}$$

Umgekehrt läßt sich zeigen, daß (N1)–(N4) nicht bloß auf die euklidische Länge, sondern ebenso auf eine Reihe anderer Arten einem Vektor eine reelle, nichtnegative Zahl zuzuordnen, zutreffen. Besonders wichtig ist zum Beispiel auch

(4.4)
$$||x|| = \max_{1 \leq i \leq n} |\xi_i|.$$

Mit dieser speziellen Vektornorm assoziieren wir für $A \in \mathbb{K}^{n\times n}$ die **Matrixnorm**:

$$||A|| = max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}|. \tag{4.5}$$

Man überzeuge sich davon, daß auch $||A||$ die Axiome einer Norm (N1) – (N4) erfüllt und noch zusätzlich folgende Ungleichungen gelten, welche dem Abbildungscharakter von A Rechnung tragen:

$$||AB|| \le ||A||\,||B|| \tag{4.6}$$

$$||Ax|| \le ||A||\,||x|| \qquad \text{Kompatibilität mit Vektornorm} \tag{4.7}$$

Wir ziehen die Vektornorm (4.4) der euklidischen Norm an dieser Stelle vor, da die euklidischen Norm auf eine wesentlich schwieriger zu berechnende Matrixnorm führt.

Beispiel 4.3. Die Norm der Koeffizientenmatrix von (4.1) ergibt sich aus

$$|a_{11}| + |a_{12}| = 1 + 11 = 12 \qquad |a_{21}| + |a_{22}| = 9 + 100 = 109,$$

also

$$||A|| = \max\{12, 109\} = 109. \tag{4.8}$$

Wir betrachten nun 2 lineare Gleichungssyteme mit gleicher Koeffizientenmatrix

$$Ax = b, \qquad A\hat{x} = \hat{b}.$$

Man kann sich $\hat{b}$ als fehlerbehaftete Realisierung einer "wahren" Inhomogenität b vorstellen, beispielsweise kann b durch Rundungsfehler beim Abspeichern am PC und/oder durch Meßfehler verfälscht werden. Ferner nehmen wir an, daß A regulär ist. Die jeweiligen Lösungen bezeichnen wir mit x und $\hat{x}$. Subtraktion der beiden Gleichungen ergibt

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

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

Wegen der Kompatibilität der Vektor- mit der Matrixnorm (4.7) folgt weiter

$$||x - \hat{x}|| = ||A^{-1}(b - \hat{b})|| \le ||A^{-1}||\,||b - \hat{b}||,$$

dividiert man nun noch durch $||x||$ ($||x|| \neq 0$ falls $b \neq 0$), erhält man

$$\frac{||x - \hat{x}||}{||x||} \le \frac{||A^{-1}||\,||b - \hat{b}||}{||x||} = ||A||\,||A^{-1}||\frac{||b - \hat{b}||}{||A||||x||}. \tag{4.9}$$

Schließlich verwenden wir noch einmal die Gleichung $Ax = b$ und (4.7),

$$||b|| = ||Ax|| \le ||A||\,||x||,$$

also

$$\frac{1}{||A||\,||x||} \le \frac{1}{||b||}, \tag{4.10}$$

und setzen (4.10) in (4.9) ein. Dies führt auf

$$\frac{||x-\hat{x}||}{||x||} \leq ||A||\,||A^{-1}||\frac{||b-\hat{b}||}{||b||}. \tag{4.11}$$

Dieses wichtige Störungsresultat sagt aus, daß der relative Fehler in b schlimmstenfalls mit dem Faktor $||A||\,||A^{-1}||$ verstärkt wird. Die Abschätzung (4.11) ist übrigens scharf: zu einem gegebenen Gleichungssystem kann man nämlich Störungen $\hat{b}$ angeben, sodaß in (4.11) Gleichheit steht. Man nennt den Verstärkungsfaktor

$$\kappa(A) = ||A||\,||A^{-1}|| \tag{4.12}$$

Konditionszahl der Inversion. Aus $||E|| = 1$ und $||E|| = ||AA^{-1}|| \leq ||A||\,||A^{-1}||$ (wegen (4.6)) folgert man

$$\kappa(A) \geq 1.$$

Verwendet man also die Konditionszahl der Inversion, ist eine Matrix perfekt konditioniert für $\kappa(A) = 1$ und schlecht konditioniert für $\kappa(A) \gg 1$.

Beispiel 4.4. Die Inverse der Koeffizientenmatrix in (4.2) ist

$$A^{-1} = \begin{pmatrix} 100 & 11 \\ 9 & 1 \end{pmatrix},$$

also $||A^{-1}|| = \max\{111, 10\} = 111$. Mit (4.8) ergibt sich die Konditionszahl

$$\kappa(A) = 111 \cdot 109 = 12099.$$

Wendet man (4.11) auf (4.1) an, erhält man die a priori Abschätzung

$$\frac{||y-\hat{y}||}{||y||} \leq 12099\frac{||d-\hat{d}||}{||d||} = 12099\frac{1}{109} = 111.$$

Dies stimmt recht gut mit der tatsächlich beobachteten relativen Abweichung in y überein:

$$\frac{||y-\hat{y}||}{||y||} = \frac{||\begin{pmatrix} 100 \\ 9 \end{pmatrix}||}{||\begin{pmatrix} -1 \\ 1 \end{pmatrix}||} = 100.$$

4.2. Bewertung der numerischen Lösung. Unsere bisherige Diskussion hat gezeigt, daß eine gewisse Skepsis gegenüber der numerischen Lösung eines linearen Gleichungssystems angebracht sein kann. Es ist naheliegend, die Genauigkeit einer "Lösung" dadurch zu überprüfen, daß man nachsieht, wie gut die Gleichung tatsächlich erfüllt wird. Es sei beispielsweise $\hat{x}$ die berechnete Lösung von $Ax = b$. Man setzt nun $\hat{x}$ in die Gleichung ein und bildet den **Residuenvektor**

$$r = b - A\hat{x}. \tag{4.13}$$

Wäre $\hat{x}$ die exakte Lösung, müßte natürlich $r = 0$ gelten. Wir vermuten daher, daß $\hat{x}$ dann eine gute Approximation der exakten Lösung darstellt, wenn $||r||$ klein ist.

BEISPIEL 4.5. Wir testen diese Hypothese an folgendem Beispiel:

$$\begin{aligned} 0.780x_1 + 0.563x_2 &= 0.217 \\ 0.913x_1 + 0.659x_2 &= 0.254. \end{aligned}$$

Mit Hilfe verschiedener Verfahren seien die Näherungslösungen

$$\hat{x} = \begin{pmatrix} 0.341 \\ -0.087 \end{pmatrix} \quad \text{und } \hat{y} = \begin{pmatrix} 0.999 \\ -1.001 \end{pmatrix}$$

gewonnen worden. Die zugehörigen Residuenvektoren lauten

$$r_{\hat{x}} \doteq \begin{pmatrix} 10^{-6} \\ 0 \end{pmatrix} \quad \text{und } r_{\hat{y}} \doteq \begin{pmatrix} 0.0013 \\ -0.0015 \end{pmatrix}.$$

Auf Grund obiger Vermutung würde man wohl $\hat{x}$ als Lösung akzeptieren. Die wahre Lösung des Systems ist allerdings

$$x = \begin{pmatrix} 1 \\ -1 \end{pmatrix}!$$

Die Ursache dieses Fehlschlages besteht darin, daß wir die schlechte Kondition des Gleichungssystems

$$\kappa(A) = 2.66 \cdot 10^6$$

nicht berücksichtigt haben.

Welche Information kann man aus dem Residuenvektor herausholen? Für reguläres A ist wegen

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

der Fehler der Näherungslösung

$$e := x - \hat{x}$$

die Lösung des Systems

$$Ae = r, \tag{4.14}$$

und führt zu der Abschätzung

$$||e|| \leq ||A^{-1}||\,||r|| = \kappa(A)\frac{||r||}{||A||}. \tag{4.15}$$

Ist die Konditionszahl also nicht zu groß, ist die Norm des Residuenvektors ein zuverlässiger Indikator für die Genauigkeit der berechneten Lösung. Multipliziert man die Gleichung $Ax = b$ mit einem Skalar ungleich Null, bleibt zwar die Lösung unverändert, der Residuenvektor jedoch wird ebenfalls mit dem Skalar multipliziert. Auf diese Weise kann die Norm des Residuenvektor beliebig klein gemacht werden.

Daher ist irgendeine Form der Skalierung des Gleichungsfehlers wie beispielsweise in in (4.15) notwendig.

5. Iterative Lösungsmethoden

Das Eliminationsverfahren nach Gauß ist der Prototyp eines **direkten** Verfahrens zur Lösung eines linearen Gleichungssystems. Direkte Verfahren führen nach einer endlichen Anzahl von Schritten zur exakten Lösung (natürlich nur, wenn sämtliche Zwischenrechnungen exakt sind). Der Nachteil dieser Verfahren ist, daß sie im allgemeinen mit einem großen Rechenaufwand verbunden sind. Für sehr hochdimensionale Probleme, welche in den Anwendungen sehr häufig auftreten, kann der Rechenaufwand leicht die Kapazität moderner Computer sprengen. Allerdings ist es in diesen Fällen oft möglich, zumindest die Produkte Ax auszuführen. In diesen Fällen kann der Einsatz eines *iterativen* Verfahrens versucht werden. Deren Grundidee besteht darin, die lineare Gleichung als Fixpunktgleichung zu schreiben, welche iterativ gelöst wird. Wir können hier nur die zwei einfachsten Verfahren beschreiben. Dazu betrachten wir wieder ein lineares Gleichungssystem

$$Ax = b$$

mit invertierbarer Koeffizientenmatrix A. Ferner setzen wir $a_{ii} \neq 0$, $i = 1, \ldots, n$ voraus. Faßt man die Diagonalelemente in einer Diagonalmatrix D zusammen, d.h. $d_{ii} = a_{ii}$, $i = 1, \ldots, n$, und $d_{ij} = 0$, $i \neq j$, und alle übrigen Elemente von A in der Matrix N, kann man A aufspalten in

$$A = D + N. \tag{5.1}$$

Nach Voraussetzung ist D invertierbar. Somit ist das lineare Gleichungssystem

$$Dx + Nx = b$$

gleichwertig mit

$$x = D^{-1}(b - Nx). \tag{5.2}$$

Setzt man $d = D^{-1}b$ und $M = -D^{-1}N$, erhält man die allgemeine Fixpunktgleichung

$$x = d + Mx. \tag{5.3}$$

Ausgehend von einem Startwert $x^{(0)}$ kann man versuchen (5.2) mit der Methode der sukkzessiven Approximationen (4.1) zu lösen:

$$x^{(k+1)} = D^{-1}(b - Nx^{(k)}), \qquad k = 0, 1, \ldots \tag{5.4}$$

Ausführlich ausgeschrieben bedeutet (5.4):

Algorithmus 5.1 (Jacobi Verfahren). *Es sei $A \in \mathbb{K}^{n\times n}$ invertierbar und $a_{ii} \neq 0$, $i = 1, \ldots, n$. Ferner sei $tol > 0$ eine gegebene Fehlertoleranz.*

Schritt 0: *Wähle Startwert* $x^{(0)}$
Schritt k: *Es steht* $x^{(k)}$ *zur Verfügung*
Berechne für $i = 1, \ldots, n$

$$x_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)}) \tag{5.5}$$

Teste Abbruchbedingung

$$||x^{(k+1)} - x^{(k)}|| < tol.$$

In der Abbruchbedingung kann die Norm (4.4) verwendet werden.

Beispiel 5.1. Wir demonstrieren das Jacobi Verfahren an folgendem 3×3 System:

$$\begin{aligned} 6x_1 + 2x_2 + x_3 &= -11 \\ 3x_2 + x_3 &= 2 \\ x_1 + 2x_2 + 5x_3 &= -5, \end{aligned}$$

welches die exakte Lösung

$$x_1 = -2 \quad x_2 = 1 \quad x_3 = -1$$

Tabelle 5.1. Jacobi Iterationen für Beispiel 5.1

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$\|\|x - x^{(k)}\|\|$	$\|\|x^{(k)} - x^{(k-1)}\|\|$	$\frac{\|\|x^{(k)} - x^{(k-1)}\|\|}{\|\|x^{(k-1)} - x^{(k-2)}\|\|}$
0	0.0000	0.0000	0.0000	2.0000		
1	−1.8333	0.6667	−1.0000	0.3333	1.8333	
2	−1.8889	1.0000	−0.9000	0.1111	0.3333	0.182
3	−2.0167	0.9667	−1.0222	0.0333	0.1278	0.383
4	−1.9852	1.0074	−0.9833	0.0167	0.0407	0.319
5	−2.0052	0.9944	−1.0059	0.0059	0.0226	0.555
6	−1.9972	1.0020	−0.9967	0.0033	0.0092	0.407
7	−2.0012	0.9989	−1.0014	0.0014	0.0046	0.503
8	−1.9994	1.0005	−0.9993	0.0007	0.0020	0.440
9	−2.0003	0.9998	−1.0003	0.0003	0.0010	0.480
10	−1.9999	1.0001	−0.9999	0.0001	0.0004	0.453
11	−2.0001	1.0000	−1.0001	0.0001	0.0002	0.470
12	−2.0000	1.0000	−1.0000	0.0000	0.0001	0.459

besitzt. In diesem Falle ist

$$D = \begin{pmatrix} 6 & & \\ & 3 & \\ & & 5 \end{pmatrix} \quad \text{und} \quad N = \begin{pmatrix} 0 & 2 & 1 \\ 0 & 0 & 1 \\ 1 & 2 & 0 \end{pmatrix}.$$

Aus Mangel an weiteren Informationen über die Lösung wählen wir $x_0 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$ als Startwert. Die ersten zwölf Iterationen sind in Tabelle 5.1 zusammengefaßt. Die Iteration wurde abgebrochen bei Erreichen von $||x^{(k)} - x^{(k-1)}|| < 10^{-3}$ (Spalte 6 in Tabelle 5.1). Der letzten Spalte entnehmen wir, daß die Jacobi Iterationen linear konvergieren (in Übereinstimung mit unserer allgemeinen Diskussion von Fixpunktiterationen in Abschnitt 4).

Können wir die Konvergenz der Iterationen beschleunigen? Einen Hinweis gibt die detaillierte Iterationsvorschrift (5.5). Wir gehen doch von der Vorstellung aus, daß $x^{(k+1)}$ im allgemeinen eine bessere Approximation der Lösung darstellt als $x^{(k)}$. Bei der Berechnung von $x_i^{(k+1)}$ stehen bereits die Koordinaten $x_j^{(k+1)}$, $j = 1, \ldots, i-1$, zur Verfügung, sie werden aber nicht verwendet. Vielleicht können wir die Approximation verbessern, wenn wir für die Berechnung von $x_i^{(k+1)}$ bereits die Koordinaten $x_j^{(k+1)}$, $j = 1, \ldots, i-1$, einsetzen. Dies führt auf folgenden Algorithmus:

ALGORITHMUS 5.2 (Gauß-Seidel Verfahren). *Es sei $A \in \mathbb{K}^{n\times n}$ invertierbar und $a_{ii} \neq 0$, $i = 1, \ldots, n$. Ferner sei $tol > 0$ eine gegebene Fehlertoleranz.*

Schritt 0: *Wähle Startwert x_0*

Schritt k: *Es steht $x^{(k)}$ zur Verfügung*
Berechne für $i = 1, \ldots, n$

$$x_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}) \tag{5.6}$$

Teste Abbruchbedingung

$$||x^{(k+1)} - x^{(k)}|| < tol.$$

Tabelle 5.2 zeigt die Gauß–Seidel Iterationen für Beispiel 5.1. Zumindest für dieses Beispiel konvergiert das Gauß–Seidel Verfahren tatsächlich schneller als das Jacobi Verfahren. Allgemein kann man beweisen, daß das Gauß–Seidel Verfahren konvergiert, wenn das Jacobi Verfahren konvergiert (und dann sogar rascher als das Jakobi Verfahren).

Auch das Gauß–Seidel Verfahren läßt sich in das allgemeine Schema 5.3 einordnen: Dazu betrachtet man die Aufspaltung

$$A = D + L + U,$$

wobei L die Matrixelemente von A unterhalb, U jene oberhalb der Hauptdiagonale erfaßt. Die Iterationsvorschrift (5.6) kann nun kompakt dargestellt werden in der

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$\|\|x - x^{(k)}\|\|$	$\|\|x^{(k)} - x^{(k-1)}\|\|$	$\frac{\|\|x^{(k)}-x^{(k-1)}\|\|}{\|\|x^{(k-1)}-x^{(k-2)}\|\|}$
0	0.0000	0.0000	0.0000	2.0000		
1	−1.8333	0.6667	−0.9000	0.3333	1.8333	
2	−1.9056	0.9667	−1.0056	0.0944	0.3000	0.164
3	−1.9880	1.0019	−1.0031	0.0120	0.0824	0.275
4	−2.0001	1.0010	−1.0004	0.0010	0.0121	0.147
5	−2.0003	1.0001	−1.0000	0.0003	0.0009	0.076
6	−2.0000	1.0000	−1.0000	0.0000	0.0002	0.260
7	−2.0000	1.0000	−1.0000	0.0000	0.0000	0.184

TABELLE 5.2. Gauß-Seidel Iterationen für Beispiel 5.1

Form

$$Dx^{(k+1)} = b - Lx^{(k+1)} - Ux^{(k)}. \tag{5.7}$$

Die Matrix $D + L$ ist eine untere Dreiecksmatrix, deren Diagonalelemente a_{ii} nicht Null sind. Somit ist $D + L$ invertierbar und (5.7) ist gleichwertig mit

$$x^{(k+1)} = (D + L)^{-1}(b - Ux^{(k)}). \tag{5.8}$$

Setzt man $d = (D + L)^{-1}b$ und $M = -(D + L)^{-1}U$ findet man die Struktur (5.3) der Fixpunktgleichung wieder.

BEISPIEL 5.2. Wir betrachten noch einmal das Gleichungssystem 5.1. Allerdings vertauschen wir die Reihenfolge der Unbekannten x_1 und x_3 und jene der ersten und zweiten Gleichung. Setzt man $u = x_3$, $v = x_2$ und $w = x_1$, ergibt sich das äquivalente Gleichungssystem

$$\begin{array}{rcr} u + 3v & = & 2 \\ u + 2v + 6w & = & -11 \\ 5u + 2v + w & = & -5 \end{array}$$

Abbildung 5.1 zeigt die Jacobi Iterationen für x_1 durchgezogen, x_2 strichliert und x_3 punktiert mit demStartwert $x_0 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$. Offensichtlich divergieren die Iterationen (es divergieren auch die entsprechenden Gauß–Seidel Iterationen). Was ist der Grund für das unterschiedliche Verhalten der Algorithmen bei diesen äquivalenten Gleichungssystemen?

Dazu betrachten wir die allgemeine Fixpunktgleichung (5.3) und die Iterationen

$$x^{(k+1)} = d + Mx^{(k)}. \tag{5.9}$$

Es sei x^* der Fixpunkt von (5.3) und

$$e^{(k)} = x^* - x^{(k)}$$

der Fehler in der k–ten Iteration. Mit Hilfe der Fixpunktgleichung und der Iterationsvorschrift findet man

$$e^{(k+1)} = (d + Mx^*) - (d + Mx^{(k)}) = Mx^* - Mx^{(k)} = M(x^* - x^{(k)}) = Me^{(k)},$$

und damit die Abschätzung

$$||e^{(k+1)}|| \leq ||M||\,||e^{(k)}||.$$

Ein einfaches Induktionsargument ergibt schließlich die Fehlerabschätzung

$$||e^{(k)}|| \leq ||M||^k ||x^* - x^{(0)}||, \tag{5.10}$$

aus der man

$$||M|| < 1 \tag{5.11}$$

als hinreichende Konvergenzbedingung abliest. Wir fassen diese Diskussion zusammen:

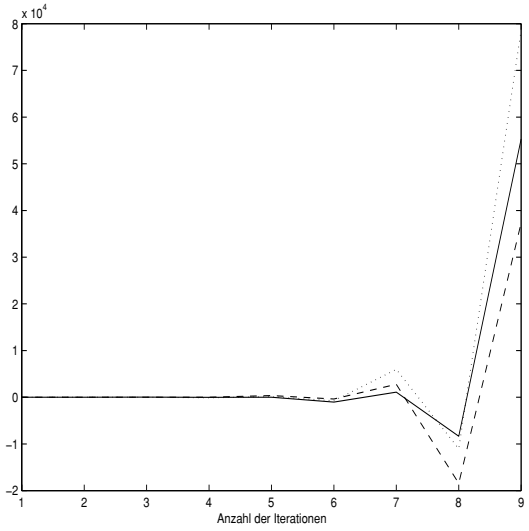
THEOREM 5.1. *Die Iterationen* (5.9) *der Fixpunktgleichung* (5.3) *konvergieren unabhängig von der Wahl des Startwertes falls* (5.11) *gilt.*

BEISPIEL 5.3. In Beispiel 5.1 erhält man

$$M_1 = -\begin{pmatrix} 0 & \frac{2}{6} & \frac{1}{6} \\ 0 & 0 & \frac{1}{3} \\ \frac{1}{5} & \frac{2}{5} & 0 \end{pmatrix}$$

und somit $||M_1|| = \frac{3}{5}$. Dies stimmt auch ungefähr mit der beobachteten Konvergenzgeschwindigkeit überein. Die Iterationsmatrix M_2 für Beispiel 5.2 ist

ABB. 5.1. Jacobi Iterationen für Beispiel 5.2



$$M_2 = \begin{pmatrix} 0 & 3 & 0 \\ \frac{1}{2} & 0 & \frac{6}{2} \\ 5 & 2 & 0 \end{pmatrix}$$

und daher $||M_2|| = 7$. Konvergenz der Iterationen ist daher nicht zu erwarten.

KAPITEL 4

Interpolation und Approximation

Noch vor wenigen Jahrzehnten erfolgte die Auswertung der elementaren Funktionen mit Hilfe von Tafeln. In einer Logarithmentafel beispielsweise sind die dekadischen Logarithmen der reellen Zahlen zwischen 1 und 10 in Schritten von $\frac{1}{1000}$ festgehalten. Die Logarithmen von nicht tabellierten Zahlen wurden durch **Interpolation** aus den Tabellenwerten errechnet. Wegen der enormen praktischen Bedeutung wurden im Laufe der Jahrhunderte die verschiedensten Verfahren entwickelt, um den erforderlichen, händischen Rechenaufwand zu minimieren. Obwohl Taschenrechner und moderne Computer die meisten derartigen Resultate auf Knopfdruck bereitstellen, gibt es in Theorie und Anwendungen nach wie vor Bedarf an Interpolationsmethoden.

Ein anderes häufig auftretendes Problem besteht darin, einen diskreten Datensatz oder eine Funktion durch eine glatte, einfach auszuwertende Funktion "möglichst gut" zu approximieren. Für die Lösung dieser Probleme stellen wir Methode der kleinsten Quadrate vor.

1. Interpolation

1.1. Die Auswertung eines Polynoms. Eine Polynomfunktion

$$p(x) = \sum_{i=0}^{n} a_i x^i \tag{1.1}$$

kann auf verschiedene Arten ausgewertet werden. Am ineffizientesten ist folgendes Programmfragment, in welchem wir annehmen, daß die Koeffizienten in einem Vektor a mit $n+1$ Elementen abgespeichert sind:

```
poly=a(1);
for i= 1:n
poly = poly + a(i+1)*x^i;
end
```

Jeder Term $a_i x^i$ erfordert nämlich i Multiplikationen, für die Auswertung des gesamten Polynoms (1.1) sind also

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Multiplikationen durchzuführen. Im Gegensatz zu vielen anderen Paketen ist es in MATLAB (derzeit) nicht möglich, Variable mit dem Index 0 zu versehen, weshalb wir in diesem Fragment $a(1), \ldots, a(n+1)$ für $a_0, \ldots, a_n$ programmiert haben.

Besser ist es, den rekursiven Charakter der Potenz zu berücksichtigen:

$$x^i = x^{i-1}x,$$

wie beispielsweise in der folgenden Modifikation

```
poly=a(1);
potenz = 1;
for i= 1:n
potenz = potenz*x;
poly = poly + a(i+1)*potenz;
end
```

Offensichtlich sind für die Auswertung von $a_i x^i$, $i > 1$, jeweils nur mehr 2 Multiplikationen notwendig. Dies ergibt insgesamt

$$1 + 2(n-1)$$

Multiplikationen. Der Rechenaufwand läßt sich durch Schachtelung der Multiplikationen noch weiter drücken. Dazu schreibt man das Polynom in der Form

$$p(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + a_n x)\ldots).$$

Die Auswertung des Polynoms in dieser geschachtelten Form führt auf folgenden Algorithmus:

ALGORITHMUS 1.1 (Hornersches Schema). *Für die Auswertung des Polynoms* (1.1) *an der Stelle ξ berechne man iterativ die Zahlen*

$$\begin{aligned} b_n &= a_n \\ b_{n-1} &= a_{n-1} + b_n\xi \\ b_{n-2} &= a_{n-2} + b_{n-1}\xi \\ &\vdots \\ b_0 &= a_0 + b_1\xi. \end{aligned}$$

Dann gilt

$$p(\xi) = b_0$$

Das Hornersche Schema benötigt nur mehr n Multiplikationen für die Berechnung von $p(\xi)$.

1.2. Lagrange Interpolation. Gegeben seien Daten der Form (x_i, y_i), $i = 0, \ldots, n$, mit paarweise verschiedenen **Stützstellen** x_i. Beispielsweise könnten y_i tabellierte Werte einer Funktion für verschiedene Argumente x_i, oder Meßwerte einer physikalischen Größe für unterschiedliche Werte eines Parameters x_i repräsentieren.

Gesucht ist ein Polynom $p(x) = \sum_{i=0}^{k} a_i x^i$ minimalen Grades, welches die Interpolationsbedingungen

$$p(x_j) = y_j, \quad j = 0, \ldots, n, \tag{1.2}$$

erfüllt. Dies führt auf das lineare Gleichungssystem

$$\sum_{i=0}^{k} a_i x_j^i = y_j, \quad j = 0, \ldots, n, \tag{1.3}$$

für die unbekannten Koeffizienten a_i, $i = 0, \ldots, k$, des interpolierenden Polynoms. Aus der Theorie der linearen Gleichungssysteme ist bekannt, daß für jede Wahl von y_i eine eindeutige Lösung nur dann existieren kann, falls $k = n$ gilt. Damit ist der Grad des interpolierenden Polynoms festgelegt. Das Gleichungssystem (1.2) hat somit die Form

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}. \tag{1.4}$$

Die Koeffizientenmatrix in (1.4) heißt **Vandermonde Matrix**. Unabhängig von (1.4) kann man sich die Eindeutigkeit des interpolierenden Polynoms (soferne es existiert) überlegen. Denn gäbe es ein weiteres Polynom $\tilde{p}$ n–ten Grades, welches ebenfalls die Interpolationsbedingungen (1.2) erfüllt, müßte die Differenz $p - \tilde{p}$ die Gleichungen

$$p(x_j) - \tilde{p}(x_j) = y_j - y_j = 0, \quad j = 0, \ldots, n, \tag{1.5}$$

erfüllen. $p - \tilde{p}$ ist ein Polynom, dessen Grad nicht größer als n sein kann. Nach dem Fundamentalsatz der Algebra kann $p - \tilde{p}$ daher höchstens n Nullstellen besitzen oder es gilt $p - \tilde{p} \equiv 0$. Die Gleichungen (1.5) bedeuten, daß $p - \tilde{p}$ mindestens $n + 1$ Nullstellen hat, somit folgt $p \equiv \tilde{p}$. Man kann zeigen, daß eine Vandermonde Matrix regulär ist, soferne die Stützstellen x_i paarweise verschieden sind. Zusammenfassend wurde gezeigt:

THEOREM 1.1. *Gegeben seien Daten der Form (x_i, y_i), $i = 0, \ldots, n$, mit paarweise verschiedenen* **Stützstellen** *x_i. Dann gibt es genau ein Polynom n–ten Grades, welches die Interpolationsbedingungen* (1.2) *erfüllt. Schreibt man das interpolierende Polynom in der Form $p(x) = \sum_{i=0}^{n} a_i x^i$, erhält man die Koeffizienten als Lösungen des linearen Gleichungssystems* (1.4).

Interpoliert man eine hinreichend glatte Funktion, kann man den Interpolationsfehler folgendermaßen abschätzen:

THEOREM 1.2. *Es sei $f\colon I \to \mathbb{R}$ eine $n+1$ mal stetig differenzierbare Funktion und p_n das Polynom, welches f in $x_0 < x_1 < \cdots < x_n$ interpoliert, $x_i \in I$, $i =$*

$0, \ldots, n$. *Dann gilt für jedes* $x \in [x_0, x_n]$

$$f(x) - p_n(x) = f^{(n+1)}(\xi)\frac{(x-x_0)(x-x_1)\cdots(x-x_n)}{(n+1)!}, \tag{1.6}$$

wobei ξ *eine von* x *abhängige Zwischenstelle im Intervall* (x_0, x_n) *ist.*

Man beachte die große Ähnlichkeit von (1.6) mit dem Restglied der Taylorentwicklung, vgl Satz 3.1.

Beispiel 1.1. In der Statistik wird die normierte kumulative Verteilungsfunktion Φ benötigt, welche durch das uneigentliche Integral

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2}\, dt$$

für alle $x \in \mathbb{R}$ definiert ist. Dieses Integral kann man nicht analytisch auswerten. Daher werden gerundete Werte von $\Phi(x)$ in Tabellen zur Verfügung gestellt, wie etwa in folgendem Ausschnitt:

x	0	0.2	0.4	0.6	0.8	1.0	1.2
$\Phi(x)$	0.5	0.57926	0.65542	0.72575	0.78814	0.84134	0.88493

Es soll z.B. $\Phi(0.32)$ berechnet und der Approximationsfehler abgeschätzt werden. Wir verwenden zuerst lineare Interpolation. Wir suchen also ein Polynom $p_1(x) = a_0 + a_1 x$ mit

$$p_1(0.2) = a_0 + 0.2a_1 = 0.57926$$
$$p_1(0.4) = a_0 + 0.4a_1 = 0.65542.$$

Die eindeutige Lösung dieses Gleichungssystems ergibt

$$p_1(x) = 0.50310 + 0.38080x,$$

und somit

$$\Phi(0.32) \approx p_1(0.32) = 0.624956.$$

Bei quadratischer Interpolation macht man den Ansatz $p_2(x) = \tilde{a}_0 + \tilde{a}_1 x + \tilde{a}_2 x^2$. Als Stützstellen wählen wir $x = 0.2$, 0.4 und 0.6. Dies führt auf das Gleichungssystem

$$\begin{pmatrix} 1 & 0.2 & 0.2^2 \\ 1 & 0.4 & 0.4^2 \\ 1 & 0.6 & 0.6^2 \end{pmatrix} \begin{pmatrix} \tilde{a}_0 \\ \tilde{a}_1 \\ \tilde{a}_2 \end{pmatrix} = \begin{pmatrix} 0.57926 \\ 0.65542 \\ 0.72575 \end{pmatrix}.$$

Das quadratische Interpolationspolynom lautet somit

$$p_2(x) = 0.49727 + 0.424525x - 0.072875x^2,$$

welches als Näherungswert

$$\Phi(0.32) \approx p_2(0.32) = 0.6256556$$

ergibt. Der Interpolationsfehler kann mit Theorem 1.2 abgeschätzt werden. Dazu benötigt man die 2. bzw. 3. Ableitung von Φ. Nach dem Hauptsatz der Differential und Integralrechnung gilt für stetige Funktionen f

$$\frac{d}{dx}\int_{x_0}^{x} f(t)\,dt = f(x).$$

Man verfiziert nun leicht

$$\begin{aligned}
\Phi'(x) &= \frac{1}{\sqrt{2\pi}} e^{-x^2/2},\\
\Phi''(x) &= -\frac{1}{\sqrt{2\pi}} x e^{-x^2/2},\\
\Phi^{(3)}(x) &= -\frac{1}{\sqrt{2\pi}} (1-x^2) e^{-x^2/2},\\
\Phi^{(4)}(x) &= -\frac{1}{\sqrt{2\pi}} (-3x+x^3) e^{-x^2/2}.
\end{aligned}$$

Φ'' ist auf $[0.2, 0.4]$ negativ und monoton fallend (man achte auf das Vorzeichen von $\Phi^{(3)}$). Es folgt

$$|\Phi''(x)| \le |\Phi''(0.4)|, \quad x \in [0.2, 0.4],$$

und damit die Abschätzung

$$|\Phi(0.32) - p_1(0.32)| \le \frac{1}{2} \max_{x\in(0.2,0.4)} |\Phi''(x)| \cdot |0.32-0.2| \cdot |0.32-0.4| \approx 7.07 \cdot 10^{-4}.$$

Aus der 4. Ableitung liest man ab, daß $\Phi^{(3)}$ auf $[0.2, 0.6]$ monoton wächst,was

$$|\Phi^{(3)}(x)| \le |\Phi^{(3)}(0.2)|, \quad x \in [0.2, 0.6],$$

nach sich zieht. Bei quadratischer Interpolation erhält man also die Fehlerabschätzung

$$\begin{aligned}
|\Phi(0.32) - p_2(0.32)| &\le \frac{1}{3!} \max_{x\in(0.2,0.6)} |\Phi^{(3)}(x)| \cdot |0.32-0.2| \cdot |0.32-0.4| \cdot |0.32-0.6|\\
&\approx 1.68 \cdot 10^{-4}.
\end{aligned}$$

Direkte Auswertung von $\Phi(0.32)$ mit Hilfe numerischer Integration ergibt

$$\Phi(0.32) = 0.6255155\ldots.$$

Der jeweils tatsächliche Fehler stimmt also sehr gut mit den theoretischen Schranken überein.

Im folgenden Beispiel demonstrieren wir einige Schwierigkeiten, welche typischerweise bei polynomialer Interpolation mit äquidistanten Stützstellen auftreten.

n	$\kappa(V)$
4	135
6	2075
12	$1.39 \cdot 10^7$

TABELLE 1.1. Kondition der Vandermonde Matrix

BEISPIEL 1.2. Die Funktion $f\colon [-3,3] \to \mathbb{R}$, $f(x) = \frac{1}{1+x^2}$ soll durch Polynome approximiert werden, welche f in den Stützstellen $x_i = -3 + i\Delta x$ interpolieren. Wir wählen die Schrittweiten $\Delta x = 1.5$, 1 und 0.5. Die entsprechenden Interpolationspolynome p_4, p_6 und p_{12} sind in Abbildung 1.1 dargestellt. Man erkennt, daß mit wachsender Anzahl der Stützstellen, die Güte der Approximation in der Intervallmitte immer besser wird, an den Intervallrändern aber starke Oszillationen auftreten (**Rungephänomen**). Abbildung 1.2 zeigt den Approximationsfehler $f - p_{12}$ im Intervall $[-2,2]$. Übereinstimmend mit der Darstellung 1.2 oszilliert der Fehler (im Gegensatz zur Approximation durch Taylorpolynome) und kann daher nicht so rasch anwachsen. Man beachte, daß das interpolierende Polynom p_{12} die Funktion f gut über ein wesentlich größeres Intervall approximiert als das Taylorpolynom gleicher Ordnung (punktiert in Abbildung 1.2)

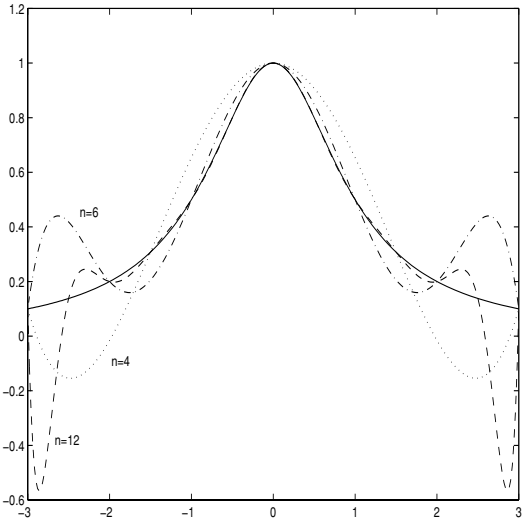


ABB. 1.1. Interpolation von $f(x) = \frac{1}{1+x^2}$

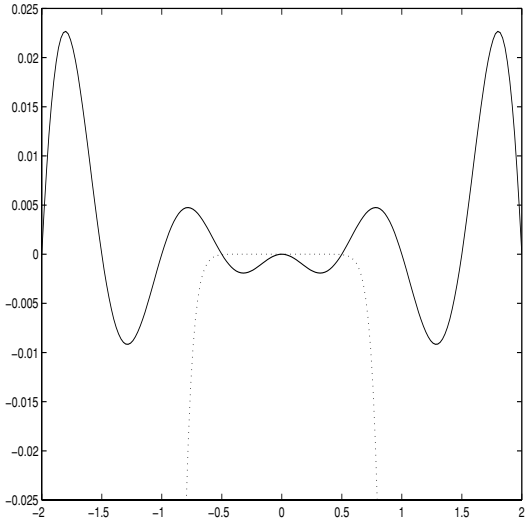


ABB. 1.2. Fehler von p_{12} und Taylorpolynom t_{12}

In Tabelle 1.1 sind die Konditionszahlen der Vandermonde Matrizen V, die in Beispiel 1.2 auftreten, zusammengefaßt. Ihr rasches Anwachsen ist ein deutlicher Hinweis für die schlechte Kondition des Gleichungssystems (1.4). Eine weitere Schwäche des bisherigen Zuganges zum Interpolationsproblem besteht darin, daß bei Hinzufügen oder Entfernen einer Stützstelle sämtliche Koeffizienten des interpolierenden Polynoms neu berechnet werden müssen. Beides kann durch eine geeignete Darstellung des Interpolanten verbessert werden. Die Randoszillationen können durch geeignete Wahl der Stützstellen gedämpft werden. Eine suboptimale Wahl soll im nächsten Abschnitt angedeutet werden.

2. Tschebyscheff Polynome

Das n–te Tschebyscheff Polynom ist für jede ganze Zahl $n \geq 0$ definiert durch

$$T_n(x) = \cos(n \arccos x), \quad -1 \leq x \leq 1. \tag{2.1}$$

Dies sieht auf den ersten Blick nicht wie ein Polynom aus, wir betrachten daher einige Speziallfälle (vgl. Abb. 2.1):

$$T_0(x) = \cos(0) = 1, \tag{2.2}$$

$$T_1(x) = \cos(\arccos x) = x, \tag{2.3}$$

$$T_2(x) = \cos(2 \arccos x) = 2\cos^2(\arccos x) - 1 = 2x^2 - 1.$$

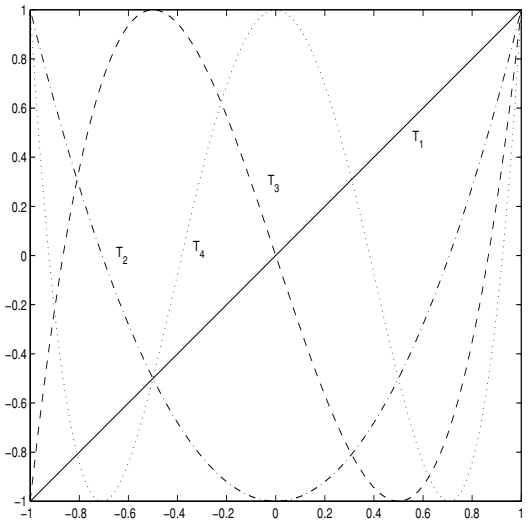


Abb. 2.1. Tschebyscheff Polynome

Aus dem Summensatz für den Kosinus leitet man leicht die Beziehung

$$\cos\alpha\cos\beta = \frac{1}{2}(\cos(\alpha+\beta)+\cos(\alpha-\beta))$$

ab. Setzt man $\theta = \arccos x$, also $x = \cos\theta$, findet man für $n \geq 1$

$$\begin{aligned} xT_n(x) &= \cos\theta\cos(n\theta) \\ &= \frac{1}{2}(\cos((n+1)\theta)+\cos((n-1)\theta) \\ &= \frac{1}{2}(T_{n+1}(x)+T_{n-1}(x)). \end{aligned} \tag{2.4}$$

Löst man (2.4) nach T_{n+1} auf, erhält man die Rekursionsformel

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n = 1,2,\ldots. \tag{2.5}$$

Zusammen mit (2.2) und (2.3) kann man die Rekursion (2.5) für die numerische Auswertung der Tschebyscheff Polynome verwenden. Mit Hilfe von (2.5) kann man sich auch leicht davon überzeugen, daß das n–te Tschebyscheff Polynom tatsächlich ein Polynom n–ten Grades der Form

$$T_n(x) = 2^{n-1}x^n + \text{Terme niedrigerer Ordnung} \tag{2.6}$$

ist. Aus der Darstellung (2.1) folgt unmittelbar die Ungleichung

$$|T_n(x)| \leq 1, \quad -1 \leq x \leq 1.$$

Der Koeffizient der höchsten Potenz des *normierten* Polynoms

$$\tilde{T}_n(x) = \frac{1}{2^{n-1}}T_n(x) \tag{2.7}$$

ist 1 und es gilt

$$\max_{x\in[-1,1]} |\tilde{T}_n(x)| = 2^{1-n}.$$

Das normierte Tschebyscheff Polynom $\tilde{T}_n$ hat eine bemerkenswerte Minimaleigenschaft:

THEOREM 2.1. *Es bezeichne $\mathcal{P}_0(n)$ die Menge der normierten Polynome vom Grad $n \geq 1$. Dann gilt*

$$2^{1-n} = \max_{x\in[-1,1]} |\tilde{T}_n(x)| = \min_{p\in\mathcal{P}_0(n)} \max_{x\in[-1,1]} |p(x)|.$$

Für stetige Funktionen $f\colon [a,b] \to \mathbb{R}$ hat

$$||f|| = \max_{x\in[a,b]} |f(x)|$$

die Eigenschaften einer Norm. Theorem 2.1 drückt daher folgendes aus: Unter allen normierten Polynomen n-ten Grades besitzt das normierte n–te Tschebyscheff Polynom minimale Norm.

2.1. Tschebyscheff Interpolation. Interpoliert man eine hinreichend glatte Funktion $f\colon [-1,1] \to \mathbb{R}$ in $x_0 < x_1 < \cdots < x_n$, gilt für den Interpolationsfehler

$$f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi)(x-x_0)(x-x_1)\cdots(x-x_n),$$

vgl. Satz 1.2. Da die Lage der Zwischenstelle ξ nicht bekannt ist, kann man den Fehler gezielt nur durch eine geeignete Wahl der Stützstellen beeinflussen: sie sollen so gewählt werden, daß

$$\max_{x\in[-1,1]} |(x-x_0)(x-x_1)\cdots(x-x_n)|$$

minimal ist. Da $(x-x_0)(x-x_1)\cdots(x-x_n)$ ein normiertes Polynom vom Grad $n+1$ ist, wird dieses Minimum nach Satz 2.1 für $\tilde{T}_{n+1}$ angenommen, als Stützstellen sind also die Nullstellen des $(n+1)$–ten Tschebyscheff Polynoms zu wählen. Diese ergeben sich aus der Gleichung

$$T_{n+1}(x) = \cos((n+1)\arccos x) = 0,$$

also

$$(n+1)\arccos x = \frac{\pi}{2} + j\pi, \quad j = 0, \pm 1, \pm 2, ...,$$

bzw.

$$x_j = \cos(\frac{2j+1}{2n+2}\pi), \quad j = 0, \ldots, n, \tag{2.8}$$

(die anderen Werte von j führen auf keine neuen Nullstellen). Interpoliert man in den Nullstellen der Tschebyscheff Polynome, spricht man von **Tschebyscheff Interpolation**. Ist die zu interpolierende Funktion auf einem Intervall $[a,b]$ gegeben, muß eine Transformation vom Intervall $[-1,1]$ auf das Intervall $[a,b]$ vorgeschalten werden:

$$z(x) = a + \frac{1}{2}(b-a)(x+1), \quad x \in [-1,1]. \tag{2.9}$$

Es gilt $z(-1) = a$ und $z(1) = b$.

Beispiel 2.1. Wir vergleichen die äquidistante Interpolation und die Tschebyscheff Interpolation für $f(x) = \frac{1}{1+x^2}$, $x \in [-3,3]$, $n = 12$. Dazu berechnen wir zuerst die Nullstellen x_i, $i = 0, \ldots, 12$,von T_{13} mit Hilfe von (2.8) und transformieren diese gemäß (2.9) auf das Intervall $[-3,3]$. Dies ergibt die Stützstellen

$$z_i = -3 + 3(x_i + 1), \quad i = 0, \ldots, 12.$$

Die beiden folgenden Abbildungen veranschaulichen die deutlich bessere globale Approximationsgüte, welche durch Tschebyscheff Interpolation erzielt werden kann. Abbildung 2.2 zeigt für $n = 12$ nochmals das Interpolationspolynom zu äquidistanten Stützstellen (strichliert) und jenes zu den Stützstellen z_i (punktiert). Auf Abbildung 2.3 kann man die entsprechenden Interpolationsfehler sehen.

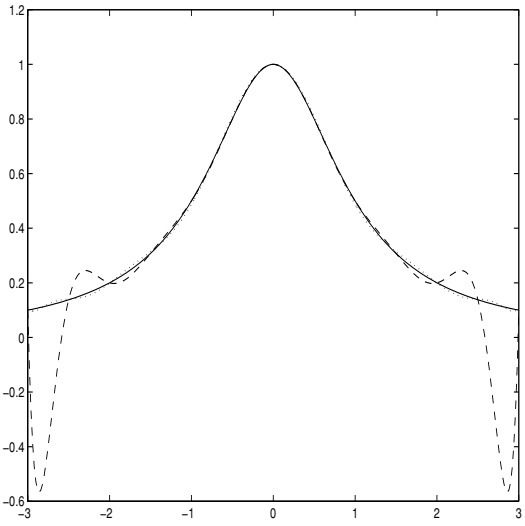


ABB. 2.2. Tschebyscheff versus äquidistante Approximation

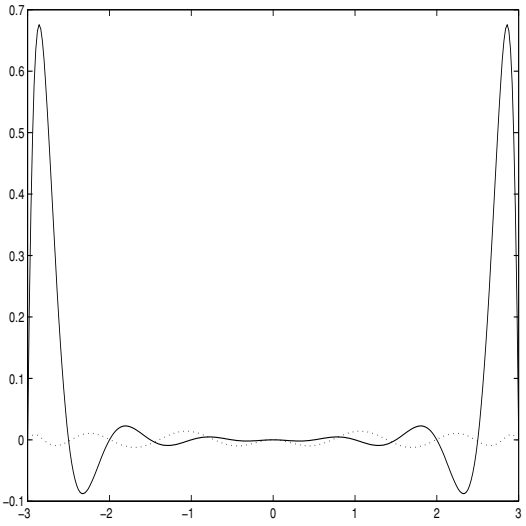


ABB. 2.3. Zugehörige Interpolationsfehler

3. Ausgleichsrechnung

BEISPIEL 3.1. Der radioaktive Zerfall einer strahlenden Substanz kann modelliert werden durch das Zerfallsgesetz

$$R(t) = R_0 e^{-\lambda t}, \quad t \geq 0. \tag{3.1}$$

Dabei bezeichnet $R(t)$ die zur Zeit t noch vorhandene Substanzmenge, R_0 die Ausgangsmenge zur Zeit $t = 0$ und λ eine materialspezifische Zerfallskonstante. Mit einem Geigerzähler werden die während einer Zeitspanne von jeweils 10 Minuten emittierten Partikel einer Indium–116 Probe gemessen: Mit Hilfe dieser Meßreihe soll die Zer-

t–Intervall	$[0,10)$	$[10,20)$	$[20,30)$	$[30,40)$	$[40,50)$	$[50,60)$
Partikel	20.511	16.174	13.904	12.514	10.775	9.569

fallskonstante für Indium–116 bestimmt werden, wobei eine Hintergrundaktivität von 632 Partikeln pro 10 Minuten zu berücksichtigen ist.

Dieses Beispiel ist ein Spezialfall der folgenden allgemeineren Problemstellung: Es sei bekannt, daß zwischen zwei interessierenden Größen y und x ein funktionaler Zusammenhang der Form

$$y = f(x)$$

existiert. Für verschiedene Werte x_i, $i = 1, \ldots, n$ mögen Meßwerte y_i zur Verfügung stehen. Die Funktion f selbst sei nicht genau bekannt, zusätzliche physikalische

(Beispiel: radioaktiver Zerfall), chemische, biologische, ökonomische oder sonstige Überlegungen oder die Form der Daten mögen allerdings die Vermutung nahelegen, daß f in einer speziellen Familie von Funktionen $\mathcal{F}$ liege (Beispiel: Zerfallsgesetz, (3.1)). Die Aufgabe besteht darin, den unbekannten funktionalen Zusammenhang so gut als möglich aus den Daten (x_i, y_i), $i = 1, \ldots, n$, zu rekonstruieren. Es liegt in der Natur des Problems, daß eine exakte "Lösung" nicht zu erwarten ist. Da die Meßwerte y_i, möglicherweise auch x_i, fehlerbehaftet sind, ist es nicht sinnvoll, f durch Interpolation zu bestimmen. Man versucht stattdessen, in der Klasse $\mathcal{F}$ eine Funktion zu finden, welche die Daten "am besten" widerspiegelt. Betrachtet man jede Funktion $f \in \mathcal{F}$ als Modell des untersuchten Vorganges, liegt es nahe, als optimales Modell jene Funktion f^* zu nehmen, für die der Modellfehler minimal ist. Als Maß für den Modellfehler kann man beispielsweise wählen unter

Absolutem Fehler:	$\sum_{i=1}^{n} \lvert f(x_i) - y_i \rvert$,
Quadratischem Fehler:	$\sum_{i=1}^{n} \lvert f(x_i) - y_i \rvert^2$,
Maximalem Fehler:	$\max\{\lvert f(x_i) - y_i \rvert,\ i = 1, \ldots, n\}$.

Wir wählen den quadratischen Fehler, denn dieser ist am einfachsten zu handhaben, weiters sprechen statistische Gründe für diese Wahl. Noch ist das Problem zu allgemein formuliert: Um zu einem brauchbaren Lösungsansatz zu kommen, nimmt man an, daß die Funktionen in der Familie $\mathcal{F}$ durch die Wahl von Parametern $\alpha_1, \ldots, \alpha_k$, $k \leq n$, eindeutig festgelegt sind, also die Form $f(x; \alpha_1, \ldots, \alpha_k)$ besitzen, und differenzierbar von den einzelnen Parametern abhängen. In Beispiel 3.1 wird man f sinnvollerweise in der Familie der Exponentialfunktionen

$$\mathcal{F} = \{\alpha_1 e^{-\alpha_2 t} : \alpha_i > 0, i = 1, 2\}$$

suchen. Jede Wahl von α_1, α_2 legt ein Zerfallsgesetz fest. Der Modellfehler

$$E(\alpha_1, \ldots, \alpha_k) = \sum_{i=1}^{n} \lvert f(x_i; \alpha_1, \ldots, \alpha_k) - y_i \rvert^2$$

kann als Funktion der Parameter $\alpha_1, \ldots, \alpha_k$ aufgefaßt werden. Die **Methode der kleinsten Quadrate** besteht nun darin, einen Parametersatz $\alpha_1^*, \ldots, \alpha_k^*$ zu finden, sodaß

$$E(\alpha_1^*, \ldots, \alpha_k^*) = \min E(\alpha_1, \ldots, \alpha_k). \tag{3.2}$$

Das Minimum ist dabei über alle zulässigen Parameter zu erstrecken. Im folgenden werden wir der Einfachheit halber voraussetzen, daß die Parameter keinen Einschränkungen unterliegen. Hängt die Familie $\mathcal{F}$ nur von einem Parameter α ab ($k = 1$), ist unter diesen Annahmen

$$\frac{d}{d\alpha} E(\alpha^*) = 0.$$

eine notwendige Optimalitätsbedingung für α^*. Betrachten wir nun 2 Parameter (α, β) und (α^*, β^*) minimiere $E(\alpha, \beta)$. Fixieren wir nun β bei einem beliebigen Wert

β_0 und betrachten die Funktion

$$h(\alpha) = E(\alpha, \beta_0).$$

Dann hat natürlich h in α^* auch ein Minimum und es muß

$$\frac{d}{d\alpha} h(\alpha^*) = 0$$

gelten. Friert man andererseits α bei einem beliebigen Wert α_0 ein, findet man analog, daß die Abbildung

$$g(\beta) = E(\alpha_0, \beta)$$

die notwendige Optimalitätsbedingung

$$\frac{d}{d\beta} g(\beta^*) = 0$$

erfüllen muß. Man nennt die Ableitungen von h und g partielle Ableitungen von E. Allgemeiner definiert man:

DEFINITION 3.1. *Es sei* $f\colon \mathbb{R}^k \to \mathbb{R}$. *Falls der Grenzwert*

$$\lim_{h\to 0} \frac{1}{h}(f(x_1, \ldots, x_{i-1}, x_i + h, x_{i+1}, \ldots, x_k) - f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_k))$$

existiert, nennt man ihn **i–te partielle Ableitung** *von* f *an der Stelle* $(x_1, \ldots, x_k)$, $i = 1, \ldots, k$. *Man schreibt*

$$\frac{\partial f}{\partial x_i}(x_1, \ldots, x_k).$$

Allgemein lautet somit eine notwendige Optimalitätsbedingung für das Optimierungsproblem (3.2)

$$\frac{\partial E}{\partial \alpha_i}(\alpha_1, \ldots, \alpha_k) = 0, \quad i = 1, \ldots, k.$$

Falls ein optimaler Parametersatz $(\alpha_1^*, \ldots, \alpha_k^*)$ existiert, ist er daher unter den Lösungen des im allgemeinen nichtlinearen Gleichungssystems

$$\sum_{i=1}^{n} (f(x_i; \alpha_1, \ldots, \alpha_k) - y_i) \frac{\partial}{\partial \alpha_j} f(x_i; \alpha_1, \ldots, \alpha_k) = 0, \quad , j = 1, \ldots, k. \tag{3.3}$$

zu finden. Die Gleichungen (3.3) nennt man **Normalgleichungen**.

BEISPIEL 3.2 (Fortsetzung). Wir suchen ein Paar von Parametern (R_0, λ), welches den Modellfehler

$$E(R_0, \lambda) = \sum_{i=1}^{6} (R_0 e^{-\lambda t_i} - y_i)^2 \tag{3.4}$$

minimiert. Als Zeitpunkte t_i nehmen wir dabei das Ende der jeweiligen Meßzeiträume, die zugehörigen Werte von y_i sind die entsprechenden Anzeigen des Geigerzählers

vermindert um die Hintergrundaktivität. Zuerst stellen wir die Normalgleichungen auf:

$$\frac{\partial E}{\partial R_0} = 2\sum_{i=1}^{6}(R_0 e^{-\lambda t_i} - y_i)e^{-\lambda t_i} = 0,$$
$$\frac{\partial E}{\partial \lambda} = 2\sum_{i=1}^{6}(R_0 e^{-\lambda t_i} - y_i)(-R_0 e^{-\lambda t_i} t_i) = 0,$$

welche nach einfachen algebraischen Umformungen geschrieben werden können als

$$\sum_{i=1}^{6} y_i e^{-\lambda t_i} - R_0 \sum_{i=1}^{6} e^{-2\lambda t_i} = 0,$$
$$\sum_{i=1}^{6} y_i t_i e^{-\lambda t_i} - R_0 \sum_{i=1}^{6} t_i e^{-2\lambda t_i} = 0.$$

Dies ist ein System nichtlinearer Gleichungen für (R_0, λ). Die erste Gleichung ergibt

$$R_0 = \frac{\sum_{i=1}^{6} y_i e^{-\lambda t_i}}{\sum_{i=1}^{6} e^{-2\lambda t_i}}. \tag{3.5}$$

Eliminiert man R_0 aus der zweiten Gleichung, erhält man als Bedingung für λ

$$\sum_{i=1}^{6} y_i t_i e^{-\lambda t_i} \sum_{i=1}^{6} e^{-2\lambda t_i} - \sum_{i=1}^{6} y_i e^{-\lambda t_i} \sum_{i=1}^{6} t_i e^{-2\lambda t_i} = 0. \tag{3.6}$$

Diese nichtlineare Gleichung lösen wir mit dem Newtonverfahren. Dazu benötigen wir einen Näherungswert für λ. Aus den Daten liest man die Beziehung

$$R(10) \approx 2R(50), \quad \text{also } R_0 e^{-10\lambda} \approx 2R_0 e^{-50\lambda}$$

ab, aus welcher man durch Logarithmieren den Schätzwert

$$\lambda \approx \frac{\log 2}{40}$$

bekommt. Nach wenigen Iterationen ergibt das Newtonverfahren die Nullstelle $\lambda^* = 0.016254$, aus (3.6) folgt dann $R_0^* = 22.547$. Das Zerfallsgesetz, welches mit der Meßreihe im Sinne der kleinsten Fehlerquadrate am besten verträglich ist, lautet also

$$R(t) = 22.547 e^{-0.016254t}.$$

Aus Abbildung 3.1 (durchgezogene Kurve) geht hervor, daß die Übereinstimmung des Zerfallsgesetzes mit den Daten (Sterne) akzeptabel ist. Dies ist nicht so leicht aus der minimalen Summe der Fehlerquadrate zu sehen:

$$E(R_0^*, \lambda^*) = 1.6 \cdot 10^6.$$

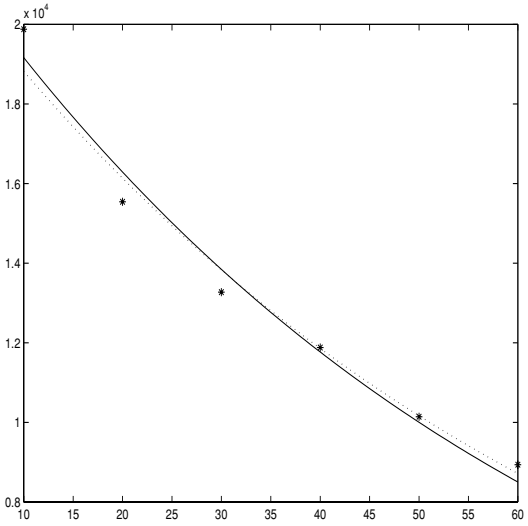


ABB. 3.1. Radioaktiver Zerfall

4. Das lineare Ausgleichsproblem

Ein **lineares Ausgleichsproblem** liegt vor, wenn die Modellfunktionen linear von den Parametern abhängen, also die spezielle Struktur

$$f(x;\alpha_1,\ldots,\alpha_k)=\sum_{i=1}^{k}\alpha_i\varphi_i(x)$$

besitzen mit vorgegebenen Funktionen φ_i, $i=1,\ldots,k$. Häufig verwendet man sogenannte **Ausgleichspolynome**

$$f(x;\alpha_1,\ldots,\alpha_k)=\sum_{i=1}^{k}\alpha_i x^{i-1}.$$

Im Falle $k=2$ spricht man von einer **Ausgleichsgeraden**. Lineare Ausgleichsprobleme werden deswegen häufig verwendet, weil die Normalengleichungen ein System linearer Gleichungen sind. Wegen

$$\frac{\partial}{\partial\alpha_j}f(x;\alpha_1,\ldots,\alpha_k)=\varphi_j(x),\quad j01,\ldots,k,$$

haben in diesem Falle die Normalengleichungen(3.3) die Form

$$\sum_{i=1}^{n}(\sum_{r=1}^{k}\alpha_r\varphi_r(x_i)-y_i)\varphi_j(x_i)=0,\quad j=1,\ldots,k$$

Vertauscht man nun noch die Reihenfolge der beiden Summationen, ergibt sich schließlich

$$\sum_{r=1}^{k} \Big[\sum_{i=1}^{n} \varphi_r(x_i)\varphi_j(x_i) \Big] \alpha_r = \sum_{i=1}^{n} \varphi_j(x_i) y_i, \quad j = 1, \ldots, k. \tag{4.1}$$

Definiert man eine Matrix $\Phi = [\Phi_1 \ldots \Phi_k] \in \mathbb{R}^{n \times k}$, deren i–te Spalte Φ_i gegeben ist durch

$$\Phi_i = (\varphi_i(x_1), \ldots, \varphi_i(x_n))^T$$

und faßt man die Parameter im Vektor $\alpha = (\alpha_1, \ldots, \alpha_n)^T$ und die Meßdaten in $y = (y_1, \ldots, y_n)^T$ zusammen, kann man (4.1) kompakt schreiben in der Form

$$\Phi^T \Phi \alpha = \Phi^T y, \tag{4.2}$$

die für eine theoretische Analyse (Existenz und Eindeutigkeit einer Lösung) besonders gut geeignet ist. Wir spezialisieren nun (4.2) auf den Fall der Ausgleichsgeraden

$$f(x; \alpha, \beta) = \alpha + \beta x, \tag{4.3}$$

die sich mit $\varphi_1 \equiv 1$ und $\varphi_2(x) = x$ in den allgemeinen Formalismus einfügt. Die Matrix Φ ist in diesem Falle

$$\Phi = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$$

und

$$\Phi^T \Phi = \begin{pmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{pmatrix}.$$

Die Parameter (α^*, β^*) der Ausgleichsgeraden erhält man also als Lösung des Gleichungssystems

$$\begin{pmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \end{pmatrix}. \tag{4.4}$$

Die minimale Fehlerquadratsumme kann in diesem Falle berechnet werden durch

$$E(\alpha^*, \beta^*) = \sum_{i=1}^{n} y_i^2 - (\alpha^* \sum_{i=1}^{n} y_i + \beta^* \sum_{i=1}^{n} x_i y_i) \tag{4.5}$$

Wegen der besonderen Einfachheit der Gleichungen für die Ausgleichsgerade versucht man oft ein nichtlineares Ausgleichsproblem durch eine geeignete Transformation zu linearisieren. Wir demonstrieren dies am Beispiel 3.1:

Beispiel 4.1. Logarithmiert man das Zerfallsgesetz (3.1), ergibt sich

$$\ln R(t) = \ln R_0 - \lambda t.$$

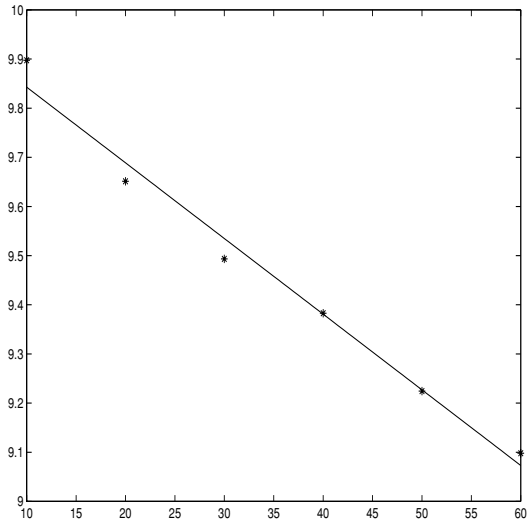


ABB. 4.1. Automatische Skalierung

Setzt man ferner $\varphi_1 \equiv 1$ und $\varphi_2(t) = -t$, erhält man ein lineares Ausgleichsproblem mit den Parametern $(\ln R_0, \lambda)$. Den Werten y_i entsprechen $\ln(R(t_i) - 632)$. Setzt man in (4.4) ein, folgt das Gleichungssystem

$$\begin{pmatrix} 6 & -210 \\ -210 & 9100 \end{pmatrix} \begin{pmatrix} \ln R_0 \\ \lambda \end{pmatrix} = \begin{pmatrix} 56.7471 \\ -1959.2180 \end{pmatrix},$$

welches durch

$$\begin{pmatrix} \ln R_0^* \\ \lambda^* \end{pmatrix} = \begin{pmatrix} 9.99672 \\ 0.015395 \end{pmatrix}$$

gelöst wird. Abbildung 4.1 ist eine *semilogarithmische Darstellung* der Daten, d.h. es wird $\ln(y_i - 632)$ versus t_i aufgetragen. Durchgezogen ist die Ausgleichsgerade

$$\ln R(t) = 9.99672 - 0.015395t$$

eingezeichnet, welche für das Auge gut mit den Daten übereinstimmt. Die minimale Fehlerquadratsumme beträgt in diesem Falle $E(\ln R_0^*, \lambda^*) = 0.0067$.

Das durch die Ausgleichsgerade bestimmte Zerfallsgesetz

$$R(t) = 21.954e^{-0.015395t}$$

ist in Abbildung 3.1 punktiert eingezeichnet. Die beiden Lösungsansätze für Beispiel 3.1 führen also zu geringfügig unterschiedlichen Ergebnissen. Beide Zerfallsgesetze stimmen jedoch gut mit den Daten überein und sind daher als gleichwertig anzusehen. Die Lösung über die Ausgleichsgerade ist jedoch wesentlich einfacher zu berechnen als über den direkten Zugang wie in Beispiel 3.2.

4.1. Beurteilung der Ausgleichsfunktion. Wir haben bereits gesehen, daß die Fehlerquadratsumme kein zuverläßiges Maß für die Güte der Approximation ist. Aber auch der visuelle Eindruck ist oft keine objektive Grundlage, zwischen verschiedenen Modellen zu wählen. Man hätte gerne eine Zahl, mit welcher die Qualität der Anpassung beurteilt werden kann. Für *lineare* Ausgleichsmodelle mit $\varphi_1 \equiv 1$ wurde mit Hilfe statistischer Methoden ein derartiges Maß entwickelt. Bezeichnet man mit $f^*(x) = \sum_{i=1}^{k} \alpha_i^* \varphi_i(x)$ das im Sinne der kleinsten Fehlerquadrate optimale Modell und mit $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$ den Mittelwert der Meßdaten, dann ist die Anpassung umso besser, je näher der Quotient

$$\rho(f^*) = \frac{\sum_{i=1}^{n} (f^*(x_i) - \bar{y})^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{4.6}$$

bei 1 liegt. Man kann zeigen, daß für $\rho(f^*)$ immer die Ungleichung

$$0 < \rho(f^*) < 1$$

gilt. Auf eine gute Datenanpassung sollte aus $\rho(f^*) \approx 1$ jedoch nur dann geschlossen werden, wenn die Anzahl der Daten jene der Funktionen φ_i um mindestens drei übertrifft, d.h. wenn

$$n - k \geq 3$$

ist.

Beispiel 4.2. Für Beispiel 4.1 findet man $\rho(f^*) = 0.98$, auf Beispiel 4.1 ist dieses Maß nicht anwendbar.

KAPITEL 5

Numerische Integration

1. Trapezregel, Simpsonregel

BEISPIEL 1.1. In einem Kolben wird das Volumen V eines Gases in Abhängigkeit vom Druck p gemessen:

p (lbf/in^2)	60.0	80.0	100	120	140	160	180
v(in^3)	80.0	69.2	60.0	52.0	45.0	38.6	32.5

Wieviel Energie ist notwendig, um das Gas von 80.0 in^3 auf 32.5 in^3 zu komprimieren?

Die Physik lehrt, daß die erforderliche Energiemenge gegeben ist durch das bestimmte Integral

$$\int_{32.5}^{80.0} p dV.$$

Eine analytische Auswertung des Integrals ist nicht möglich, da der Integrand nur in Form einer Tabelle zur Verfügung steht. Dies ist ein Beispiel für die Notwendigkeit numerischer Integrationsmethoden. Es gibt aber zahllose harmlos erscheinende Integrale, welche mit analytischen Methoden nicht exakt berechnet werden können, zum Beispiel

$$\int_0^1 e^{-x^2} dx.$$

Die grundlegende Idee hinter der approximativen Berechnung von

$$\mathcal{I}(f) = \int_a^b f(x)\, dx \tag{1.1}$$

besteht darin, den Integranden durch eine leichter zu integrierende Funktion zu approximieren. Ersetzt man f durch den linearen Interpolanten

$$p_1(x) = \frac{b-x}{b-a} f(a) + \frac{x-a}{b-a} f(b),$$

berechnet man also anstelle von $\mathcal{I}(f)$

$$\mathcal{T}_1(f) = \int_a^b p_1(x)\, dx$$

führt eine nun einfache Rechnung auf

$$\mathcal{T}_1(f) = \frac{1}{2}(b-a)[f(a)+f(b)]. \tag{1.2}$$

Geometrisch kann man $\mathcal{T}_1(f)$ als Flächeninhalt des Trapezes interpretieren, welches durch p_1 zwischen a und b definiert wird, vgl. Abbildung 1.1.

Die Näherung von $\mathcal{I}(f)$ durch $\mathcal{T}_1(f)$ ist umso genauer, je weniger der Graph von f von einer Geraden abweicht. Intuitiv wird man daher erwarten, daß der Näherungswert verbessert werden kann, indem man das Intervall $[a,b]$ in mehrere Teilintervalle unterteilt und auf jedes Teilintervall (1.2) anwendet. Der Einfachheit halber betrachten wir nur den Fall einer äquidistanten Unterteilung:

$$\begin{aligned} h &= \frac{b-a}{n} \\ x_i &= a+ih, \quad i=0,\dots,n. \end{aligned}$$

Wegen der Additivität des Integrals in bezug auf den Integrationsbereich gilt

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)\,dx.$$

Wendet man auf jedes der Teilintegrale (1.2) an, erhält man

$$\mathcal{I}(f) \approx \mathcal{T}_n(f) = \frac{h}{2}\sum_{i=1}^{n}(f(x_{i-1})+f(x_i)).$$

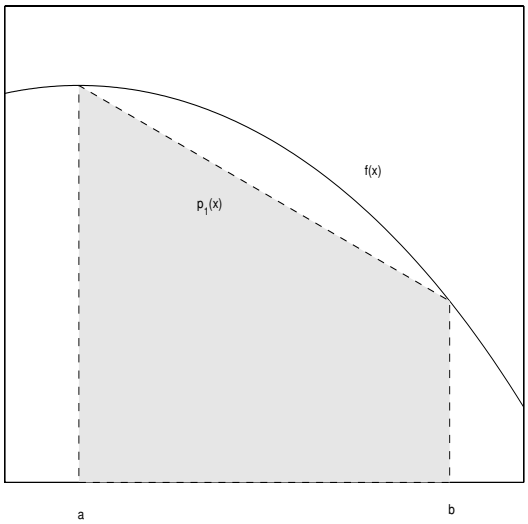


ABB. 1.1. $\mathcal{T}_1(f)$

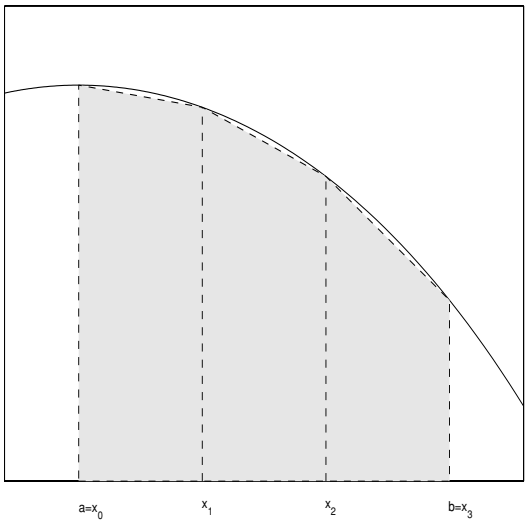


ABB. 1.2. $\mathcal{T}_3(f)$

	$\mathfrak{I}^{(1)}$		$\mathfrak{I}^{(2)}$		$\mathfrak{I}^{(3)}$	
n	e_n	$\frac{e_{n/2}}{e_n}$	e_n	$\frac{e_{n/2}}{e_n}$	e_n	$\frac{e_{n/2}}{e_n}$
1	$-5.69e-002$		$1.67e-001$		$1.53e+000$	
2	$-1.52e-002$	3.74	$6.31e-002$	2.64	$-5.61e-001$	-2.73
4	$-3.88e-003$	3.92	$2.34e-002$	2.70	$-3.76e-002$	14.93
8	$-9.75e-004$	3.98	$8.54e-003$	2.74	$-1.93e-004$	194.99
16	$-2.44e-004$	3.99	$3.09e-003$	2.77	$-5.12e-009$	37635.00
32	$-6.10e-005$	4.00	$1.11e-003$	2.79	$4.44e-016$	
64	$-1.53e-005$	4.00	$3.96e-004$	2.80	$4.44e-016$	
128	$-3.81e-006$	4.00	$1.41e-004$	2.81	$4.44e-016$	

TABELLE 1.1. Konvergenzverhalten der Trapezregel

Nach einer weiteren Vereinfachung der rechten Seite folgt schließlich die **Trapezregel:**

$$\mathfrak{T}_n(f) = h[\frac{1}{2}f(x_0) + f(x_1) + f(x_2) + \cdots + f(x_{n-1}) + \frac{1}{2}f(x_n)]. \tag{1.3}$$

Untersuchen wir die Effizienz dieses Verfahrens.

BEISPIEL 1.2. Dazu betrachten wir die Integrale:

$$\mathfrak{I}^{(1)} = \int_0^1 \frac{1}{1+x}\,dx = \ln 2,$$
$$\mathfrak{I}^{(2)} = \int_0^1 \sqrt{x}\,dx = \frac{2}{3},$$
$$\mathfrak{I}^{(3)} = \int_0^{2\pi} \frac{dx}{2+\cos x} = \frac{2\pi}{\sqrt{3}}.$$

In Tabelle 1.1 sind die **Quadraturfehler**

$$e_n = \mathfrak{I}(f) - \mathfrak{T}_n(f)$$

für wachsende Werte von n zusammengestellt. Aus rechenökonomischen Gründen wird n jeweils verdoppelt.

Die Trapezregel verhält sich bei den Funktionen in Beispiel 1.2 recht unterschiedlich: Verdoppelt man die Anzahl der Stützstellen, wird der Fehler von $\mathfrak{I}^{(1)}$ jeweils um circa den Faktor 4 reduziert, die Konvergenz von $\mathfrak{I}^{(2)}$ ist etwas langsamer, jene von $\mathfrak{I}^{(3)}$ wesentlich schneller (für $\mathfrak{I}^{(3)}$ wird bei $n = 32$ bereits Maschinengenauigkeit erreicht).

BEISPIEL 1.3 (Fortsetzung von Beispiel 1.1). Die Volumsdaten des Gases sind zwar monoton, aber nicht äquidistant. Daher ist die Trapezregel in der Form (1.3) nicht anwendbar. Die Herleitung der Trapezregel legt aber folgende Vorgangsweise nahe:

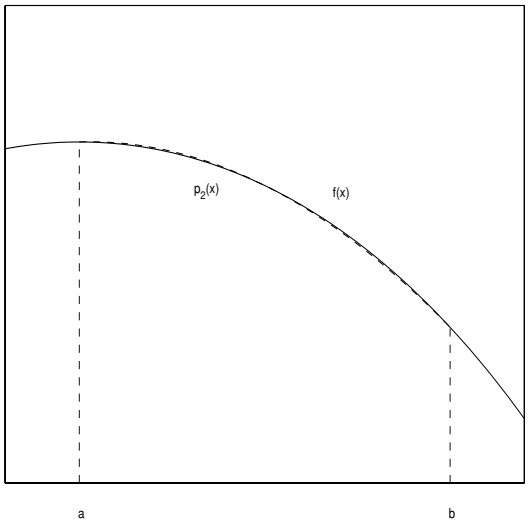


ABB. 1.3. Simpsonregel

Wir kehren zuerst die Reihenfolge der Daten um, d.h. $(V_1, p_1) = (32.5, 180)$, ..., $(V_7, p_7) = (80.0, 60.0)$, und verwenden die Näherung

$$\int_{32.5}^{80.0} pdV \approx \frac{1}{2}[(V_2 - V_1)(p_1 + p_2) + (V_3 - V_2)(p_2 + p_3) + \cdots + (V_7 - V_6)(p_6 + p_7) = 5371 \text{ lbf in.}$$

1.1. Simpsonregel. Ersetzt man den linearen Interpolanten durch den quadratischen, wird man intuitiv erwarten, daß die Näherung für (1.1) verbessert wird, vgl. Abbildung 1.3. Es sei also p_2 das Polynom 2. Grades, welches f in den Stellen $x_0 = a$, $x_1 = \frac{a+b}{2}$ und $x_2 = b$ interpoliert. Man verifiziere, daß p_2 in folgender Form dargestellt werden kann:

$$p_2(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f(x_0) + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f(x_1) + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f(x_2).$$

Als Näherungswert für $\int_a^b f(x)\,dx$ verwendet man

$$\mathcal{S}_2(f) = \int_a^b p_2(x)\,dx.$$

Eine nun einfache Integration ergibt schließlich den Grundmodul der Simpsonregel

$$\mathcal{S}_2(f) = \frac{b - a}{3}[f(a) + 4f(\frac{a + b}{2}) + f(b)]. \tag{1.4}$$

	$\mathcal{J}^{(1)}$		$\mathcal{J}^{(2)}$		$\mathcal{J}^{(3)}$	
n	e_n	$\frac{e_{n/2}}{e_n}$	e_n	$\frac{e_{n/2}}{e_n}$	e_n	$\frac{e_{n/2}}{e_n}$
2	$-1.30e-003$		$2.86e-002$		$-1.26e+000$	
4	$-1.07e-004$	12.15	$1.01e-002$	2.82	$1.37e-001$	-9.20
8	$-7.35e-006$	14.53	$3.59e-003$	2.83	$1.23e-002$	11.16
16	$-4.72e-007$	15.56	$1.27e-003$	2.83	$6.43e-005$	191.02
32	$-2.97e-008$	15.89	$4.48e-004$	2.83	$1.71e-009$	37631.01
64	$-1.86e-009$	15.97	$1.59e-004$	2.83	$-4.44e-016$	
128	$-1.16e-010$	15.99	$5.61e-005$	2.83	$-4.44e-016$	

TABELLE 1.2. Konvergenzverhalten der Simpsonregel

Wie bei der Trapezregel baut man durch Unterteilung des Integrationsbereiches eine genauere Integrationsformel zusammen. Da aber bei jedem Teilschritt für die quadratische Interpolation 3 Stützpunkte benötigt werden, faßt man je zwei benachbarte Teilintervalle zusammen. Aus diesem Grunde muß die Anzahl der Stützpunkte ungerade, bzw. n gerade sein. Bei äquidistanter Unterteilung erhält man

$$
\begin{aligned}
h &= \frac{b-a}{n}, \quad n \text{ gerade}, \\
\mathcal{S}_n(f) &= \frac{h}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \\
&\quad \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)].
\end{aligned} \tag{1.5}
$$

Diese Quadraturformel wird **Simpsonregel** genannt und war mehr als 2 Jahrhunderte lang eine der am häufigsten verwendeten numerischen Intergrationsmethoden. Tabelle 1.2 illustriert ihr Konvergenzverhalten für die Integranden in Beispiel 1.2. Die Simpsonapproximation für $\mathcal{J}^{(1)}$ und $\mathcal{J}^{(2)}$ konvergiert also wesentlich schneller als die Trapezregel, die Konvergenz gegen $\mathcal{J}^{(2)}$ ist jedoch wieder signifikant langsamer als jene gegen $\mathcal{J}^{(1)}$. Es ist auch bemerkenswert, daß bei $\mathcal{J}^{(3)}$ kein nennenswerter Unterschied zwischen beiden Approximationsmethoden zu beobachten ist.

2. Analyse der Trapezregel

Das unterschiedliche Konvergenzverhalten der Trapezregel in Beispiel 1.2 wird durch eine theoretische Analyse verständlich. Wir benötigen dazu folgendes technisches Hilfsmittel aus der Analysis:

THEOREM 2.1 (Mittelwertsatz der Integralrechnung). *Es seien f, $w\colon [a,b] \to \mathbb{R}$ stetige Funktionen und $w(x) \geq 0$ für alle $x \in [a,b]$. Dann gibt es eine Stelle $c \in [a,b]$,*

für welche

$$\int_a^b w(x)f(x)\,dx = f(c)\int_a^b w(x)\,dx$$

gilt.

Zuerst betrachten wir den Quadraturfehler auf dem Teilintervall $[x_{i-1}, x_i]$,

$$e_n^i = \int_{x_{i-1}}^{x_i} f(x)\,dx - \frac{h}{2}(f(x_{i-1}) + f(x_i)) = \int_{x_{i-1}}^{x_i} (f(x) - p_1(x))\,dx.$$

Auf die Differenz $f(x)-p_1(x)$ kann man die Fehlerabschätzung (1.6) anwenden, sofern f eine stetige 2. Ableitung besitzt, und erhält

$$e_n^i = -\frac{1}{2}\int_a^b (x-a)(b-x)f^{(2)}(\zeta_x)\,dx,$$

wobei ζ_x eine unbekannte Zwischenstelle in $[x_{i-1}, x]$ bezeichnet. Nehmen wir der Einfachheit halber an, daß diese Zwischenstelle stetig von x abhängt, kann man mit Satz 2.1

$$e_n^i = -\frac{1}{2}f^{(2)}(c_i)\int_a^b (x-a)(b-x)\,dx$$

folgern. Es steht c_i für eine wiederum unbekannte Zwischenstelle in $[x_{i-1}, x_i]$. Das verbleibende Integral läßt sich mit Hilfe der Transformation $\tau = x-a$ leicht berechnen:

$$\int_a^b (x-a)(b-x)\,dx = \int_0^h \tau(h-\tau)\,d\tau = \frac{h^3}{6}.$$

Insgesamt ergibt sich daher

$$e_n^i = -\frac{h^3}{12}f^{(2)}(c_i). \tag{2.1}$$

Summiert man e_n^i über alle Teilintervalle auf, findet man

$$e_n = \sum_{i=1}^n e_n^i = -\frac{h^3}{12}\sum_{i=1}^n f^{(2)}(c_i). \tag{2.2}$$

Da die 2. Ableitung stetig ist, nimmt $f^{(2)}$ auf $[a, b]$ Maximum und Minimum an. Es gibt also Konstante $m \le M$ mit

$$m = \min_{y\in[a,b]} f^{(2)}(y) \le f^{(2)}(x) \le M = \max_{y\in[a,b]} f^{(2)}(y), \quad \text{für alle } x \in [a, b].$$

Somit folgt die Abschätzung

$$-nM\frac{h^3}{12} \le e_n \le -nm\frac{h^3}{12},$$

bzw.

$$m \leq -\frac{12}{nh^3} e_n \leq M.$$

Da m und M dem Wertevorrat von $f^{(2)}$ angehören, schließt man mit Hilfe des Zwischenwertsatzes 2.2 auf die Existenz einer Stelle $\xi \in [a, b]$ mit

$$-\frac{12}{nh^3} e_n = f^{(2)}(\xi).$$

Berücksichtigt man nun noch $h = \frac{b-a}{n}$, erhält man:

THEOREM 2.2. *Es sei f auf $[a, b]$ zweimal stetig differenzierbar. Dann gilt die Fehlerabschätzung*

$$\int_a^b f(x)\,dx - \mathfrak{T}_n(f) = -\frac{h^2}{12}(b-a) f^{(2)}(\xi), \tag{2.3}$$

mit einer von n und f abhängigen Zwischenstelle $\xi \in [a, b]$.

Aus Satz 2.2 geht hervor, daß für hinreichend glatte Integranden eine Verdoppelung der Stützstellen, also eine Halbierung der Schrittweite h, ungefähr eine Viertelung des Quadraturfehlers zur Folge hat. "Ungefähr" nur deshalb, weil der Einfluß von $f^{(2)}(\xi)$ nicht quantifiziert wird. Diese Schlußfolgerung wird durch die numerischen Resultate für $\mathfrak{I}^{(1)}$ in Tabelle 1.1 ausgezeichnet bestätigt. Wir sind nun aber auch in der Lage, die langsame Konvergenz der Trapezregel für $f(x) = \sqrt{x}$ plausibel zu machen. Diese Funktion ist nämlich in $x_0 = 0$ *nicht* differenzierbar und erfüllt daher nicht die Voraussetzungen von Satz 2.2.

Warum aber ist die Konvergenz der Tapezregel für $\mathfrak{I}^{(3)}$ um so viel rascher? Was ist das Besondere am Integranden $f(x) = \frac{1}{2+\cos x}$? Kehren wir noch einmal zu (2.2) zurück und spalten wir einen Faktor h ab:

$$e_n = \sum_{i=1}^{n} e_n^i = -\frac{h^2}{12} \sum_{i=1}^{n} h f^{(2)}(c_i).$$

Nun läßt sich e_n als Riemannsche Summe interpretieren, welche wegen der Stetigkeit von $f^{(2)}$ einen Grenzwert besitzt:

$$\begin{aligned} \lim_{n\to\infty} e_n &= -\frac{h^2}{12} \lim_{n\to\infty} \sum_{i=1}^{n} h f^{(2)}(c_i) \\ &= -\frac{h^2}{12} \int_a^b f^{(2)}(x)\,dx = -\frac{h^2}{12}(f'(b) - f'(a)). \end{aligned}$$

Für hinreichend große n gilt demnach

$$e_n \approx \tilde{e}_n = -\frac{h^2}{12}(f'(b) - f'(a)). \tag{2.4}$$

n	e_n	$\tilde{e}_n$	$\tilde{\mathcal{T}}_n(f)$	$\mathcal{I}(f)-\tilde{\mathcal{T}}_n(f)$	Quotient
1	$-5.685e-002$	$-6.250e-002$	0.687500000000	$5.65e-003$	
2	$-1.519e-002$	$-1.562e-002$	0.692708333333	$4.39e-004$	$1.3e+001$
4	$-3.877e-003$	$-3.906e-003$	0.693117559524	$2.96e-005$	$1.5e+001$
8	$-9.747e-004$	$-9.766e-004$	0.693145287872	$1.89e-006$	$1.6e+001$
16	$-2.440e-004$	$-2.441e-004$	0.693147061583	$1.19e-007$	$1.6e+001$
32	$-6.103e-005$	$-6.104e-005$	0.693147173113	$7.45e-009$	$1.6e+001$
64	$-1.526e-005$	$-1.526e-005$	0.693147180094	$4.66e-010$	$1.6e+001$

TABELLE 2.1. Die korrigierte Trapezregel

Man nennt die Fehlerabschätzung 2.4, welche mit wachsendem n immer schärfer wird, **asymptotische Fehlerabschätzung**. Sie bestätigt einerseits die vorausgehende Analyse, daß sich der Integrationsfehler circa um den Faktor 4 reduziert, wenn man die Schrittweite h halbiert, *falls* $f'(b)-f'(a)\neq 0$. Andererseits zeigt sie, daß im Falle $f'(b)-f'(a)=0$ die Konvergenz rascher erfolgt. Wegen $f'(0)=f'(2\pi)=0$ haben wir somit zumindest eine teilweise Erklärung für die überaus rasche Konvergenz der Trapezregel bei $\mathcal{I}^{(3)}$ in Beispiel 1.2. Eine weitere Anwendung des asymptotischen Fehlerschätzers ergibt sich aus

$$\mathcal{I}(f)-\mathcal{T}_n(f)\approx\tilde{e}_n,$$

bzw.

$$\mathcal{I}(f)\approx\mathcal{T}_n(f)+\tilde{e}_n.$$

Dies führt auf die **korrigierte Trapezregel**

$$\mathcal{I}(f)\approx\tilde{\mathcal{T}}_n(f)=\mathcal{T}_n(f)-\frac{h^2}{12}(f'(b)-f'(a)). \tag{2.5}$$

Tabelle 2.1 demonstriert für $f(x)=\frac{1}{1+x}$ die Qualität der asymptotischen Fehlerabschätzung und zeigt, daß durch diese einfache Korrektur die Effizienz der Trapezregel mit jener der Simpsonregel vergleichbar wird.

3. Gauß Legendre Quadratur

Die Idee der Trapez– bzw. Simpsonregel zur Berechnung des Integrals $\int_a^b f(x)\,dx$ bestand darin, einerseits den Integrationsbereich in Teilintervalle zu unterteilen und andererseits den Integranden auf jedem Teilintervall durch ein Polynom *niedrigen* (jeweils gleichen) Grades zu interpolieren. Die resultierenden Quadraturformeln hatten die Form

$$\mathcal{G}_n(f)=\sum_{i=1}^{n} w_i f(x_i). \tag{3.1}$$

Die Genauigkeit der Näherung konnte durch Verfeinerung der Unterteilung von $[a, b]$ gesteigert werden. Für die Wahl der Stützstellen gab es keine besondere Strategie: Der Einfachheit und Übersichtlichkeit halber wurden äquidistante Stützstellen gewählt.

Wir ändern nun radikal unseren Standpunkt: Wir versuchen eine Quadraturformel der Form (3.1), also die Stützstellen x_i, $i = 1, \ldots, n$, und Gewichte w_i, $i = 1, \ldots, n$, so zu bestimmen, daß (3.1) für alle Polynome *möglichst hohen* Grades exakt ist. Setzt man $p_k(x) = x^k$, $k \in \mathbb{N} \cup \{0\}$, soll also

$$\mathcal{G}_n(p_k) = \mathcal{I}(p_k), \quad k = 0, \ldots, m, \tag{3.2}$$

für möglichst großes m gelten. Diese Bedingungen ergeben ein nichtlineares Gleichungssystem von $m+1$ Gleichungen für $2n$ Unbekannnte, welches in voller Allgemeinheit sehr schwierig zu diskutieren ist. Wir skizzieren die Vorgangsweise für den Fall $n = 2$ und $[a, b] = [-1, 1]$, die gesuchte Quadraturformel hat also die Form

$$\mathcal{G}_2(f) = w_1 f(x_1) + w_2 f(x_2).$$

Wir benötigen 4 Gleichungen für die Unbekannten w_1, w_2, x_1, x_2. Fordert man, daß $\mathcal{G}_2(p_k)$ exakt ist für $k = 0, 1, 2, 3$, ergibt sich das System

$$\begin{aligned}
\mathcal{G}_2(p_0) = \mathcal{I}(p_0) \qquad & 2 = w_1 + w_2, \\
\mathcal{G}_2(p_1) = \mathcal{I}(p_1) \qquad & 0 = w_1 x_1 + w_2 x_2, \\
\mathcal{G}_2(p_2) = \mathcal{I}(p_2) \qquad & \frac{2}{3} = w_1 x_1^2 + w_2 x_2^2, \\
\mathcal{G}_2(p_3) = \mathcal{I}(p_3) \qquad & 0 = w_1 x_1^3 + w_2 x_2^3.
\end{aligned}$$

Zuerst überlege man sich, daß keines der Gewichte Null sein kann (dies ergäbe einen Widerspruch zur 3. Gleichung). Es kann auch keine der Stützstellen Null sein. Ersetzt man in der 4. Gleichung vermöge Gleichung 2 den Ausdruck $w_1 x_1$ durch $-w_2 x_2$, ergibt sich daher

$$x_1^2 = x_2^2, \quad \text{also } |x_1| = |x_2|.$$

Die Möglichkeit $x_1 = x_2$ führt auf einen Widerspruch zu Gleichung 2. Es muß daher

$$x_1 = -x_2$$

gelten. Es folgt nun leicht

$$w_1 = w_2 = 1 \qquad \text{und } x_1 = -x_2 = \frac{\sqrt{3}}{2}.$$

Diese Lösung ist bis auf die Numerierung der Stützstellen eindeutig. Wir erhalten demnach die Quadraturformel

$$\mathcal{G}_2(f) = f(-\frac{\sqrt{3}}{2}) + f(\frac{\sqrt{3}}{2}). \tag{3.3}$$

	$\mathfrak{J}^{(1)}$		$\mathfrak{J}^{(2)}$		$\mathfrak{J}^{(3)}$	
n	e_n	$\frac{e_{n/2}}{e_n}$	e_n	$\frac{e_{n/2}}{e_n}$	e_n	$\frac{e_{n/2}}{e_n}$
2	$8.39e-004$		$-7.22e-003$		$8.23e-001$	
3	$2.55e-005$	32.94	$-2.51e-003$	2.87	$-4.30e-001$	-1.92
4	$7.63e-007$	33.40	$-1.16e-003$	2.16	$1.77e-001$	-2.43
5	$2.27e-008$	33.61	$-6.30e-004$	1.84	$-8.12e-002$	-2.17
6	$6.73e-010$	33.72	$-3.80e-004$	1.66	$3.55e-002$	-2.29
7	$1.99e-011$	33.79	$-2.46e-004$	1.54	$-1.58e-002$	-2.24
8	$5.82e-013$	34.25	$-1.69e-004$	1.46	$6.99e-003$	-2.26

TABELLE 3.1. Konvergenzverhalten der Gauß Legendre Quadratur

Ist der Integrationsbereich das Intervall $[a,b]$, benutzen wir die Substitution

$$x=\frac{1}{2}\big(b+a+t(b-a)\big),\quad t\in[-1,1],$$

und transformieren das Integral gemäß

$$\int_a^b f(x)\,dx=\frac{b-a}{2}\int_{-1}^{1} f(\frac{1}{2}(b+a+t(b-a))\,dt.$$

Auf das transformierte Integral kann (3.3) angewendet werden.

BEISPIEL 3.1. Wir betrachten $\mathfrak{J}^{(1)}$ aus Beispiel 1.2: Zuerst transformieren wir das Integral:

$$\int_0^1 \frac{1}{1+x}dx=\frac{1}{2}\int_{-1}^{1}\frac{1}{1+\frac{1}{2}(1+t)}dt,$$

mit (3.3) ergibt sich

$$\mathcal{G}_2(f)=\frac{1}{1+\frac{1}{2}(1-\frac{\sqrt{3}}{2})}+\frac{1}{1+\frac{1}{2}(1-\frac{\sqrt{3}}{2})}\doteq 0.692308.$$

Der Quadraturfehler

$$\ln 2-\mathcal{G}_2(f)\doteq 0.00084$$

ist wesentlich geringer als jener der vergleichbaren Trapezregel.

Betrachten wir nun den allgemeinen Fall:

$$\mathcal{G}_n(f)=\sum_{i=1}^{n} w_i f(x_i).$$

Mit Hilfe der Theorie der Orthogonalpolynome kann man zeigen, daß für $n\geq 2$ die Stützstellen x_i die Nullstellen des n–ten **Legendrepolynoms** sind. Die Legendre

Polynome werden nur auf dem Intervall $[-1,1]$ betrachtet. Sie können durch die zweistufige Rekursion

$$L_0(x) \equiv 1, \qquad L_1(x) = x$$
$$L_{n+1}(x) = \frac{2n+1}{n+1} x L_n(x) - \frac{n}{n+1} L_{n-1}(x)$$

berechnet werden. Man kann nachweisen, daß sämtliche Nullstellen der Legendre Polynome im Intervall $(-1,1)$ liegen. Die zugehörigen Gewichte ergeben sich aus

$$w_i = \frac{-2}{(n+1)L_{n+1}(x_i)L_n'(x_i)}, \quad i = 1,\ldots,n.$$

Die Stützstellen und Gewichte sind für gängige Werte von n tabelliert und müßen nicht jedesmal neu berechnet werden. Die Implementierung der Gauß Legendre Quadratur ist zwar etwas schwieriger, doch wird der zusätzliche Aufwand meist durch eine wesentlich raschere Konvergenz ausgeglichen. Tabelle 3.1 zeigt, daß beispielsweise $\mathcal{I}^1$ mit nur 8 Stützstellen genauer approximiert wird, als durch die Trapez–, bzw. Simpsonregel mit jeweis 128 Stützstellen. Die fehlende Differenzierbarkeit bei $\mathcal{I}^2$ macht sich wieder in der deutlich reduzierten Konvergenzgeschwindigkeit bemerkbar. Trotzdem sind die Approximationen wiederum jenen durch die Trapez– bzw. Simpsonregel überlegen. Dies liegt daran, daß bei der Gauß Legendre Quadratur im Gegensatz zur Trapez– und Simpsonregel der Integrand *nicht* an den Intervallenden ausgewertet wird. Die Approximationen für $\mathcal{I}^3$ sind vergleichsweise enttäuschend. Wahrscheinlich kann der Integrand nur schlecht durch Polynome niedrigen Grades (≤ 14) approximiert werden.