Skript zur
Numerik für Ingenieure und Naturwissenschaftler

Zentrum für Technomathematik
AG NUMERIK

basierend auf Vorlesungen gehalten von
Prof. Dr. Angelika Bunse-Gerstner im WS 96/97
Dr. Peter Benner im WS 98/99, WS 00/01
Priv.-Doz. Dr. Heike Faßbender im WS 99/00

Inhaltsverzeichnis

Kapitel 1

Ein einführendes Beispiel: Kind mit Spielzeug

Das nachfolgende Beispiel ist dem Buch *Solving Problems in Scientific Computing Using Maple and Matlab* von W. Gander und J. Hřebiček Springer Verlag, Berlin (1995), entnommen.
Nehmen wir an, ein Kind zieht ein an einer Stange befestigtes Spielzeug hinter sich her. Was für einen Weg legt das Spielzeug zurück, wenn das Kind in der Ebene herumspaziert, wenn es zum Beispiel im Kreis läuft? (Wer eine Abneigung gegen diese spielerische Vorstellung hat, möge sich eine analoge Problemstellung zum Beispiel bei der Roboterarmsteuerung oder ähnlichem selbst ausdenken.)
Um diesen Weg des Spielzeugs in Abhängigkeit vom Weg des Kindes berechnen zu können, benötigen wir ein mathematisches Modell, das diese Situation möglichst genau beschreibt und das es uns ermöglicht den unbekannten Weg des Spielzeuges zumindest näherungsweise zu berechnen. Das grundsätzliche Vorgehen wird allen Lesern vertraut sein.
Wir bezeichnen mit $(X(t), Y(t))$ die Position des Kindes zum Zeitpunkt t in der Ebene, mit $(x(t), y(t))$ die entsprechende Position des Spielzeuges. Die Situation zum Zeitpunkt t verdeutlicht die Abbildung 1.1, mit deren Hilfe wir uns die nachfolgenden drei Feststellungen klar machen können.

- Der Abstand zwischen den Punkten $(X(t), Y(t))$ und $(x(t), y(t))$ ist immer die Länge der Stange, die wir hier mit a bezeichnen. Also gilt

$$(X - x)^2 \; + \; (Y - y)^2 = a^2 \tag{1.1}$$

  [Erinnerung: Die (euklidische) Länge eines Vektors $\left[\begin{smallmatrix}\alpha\\ \beta\end{smallmatrix}\right] \in \mathbb{R}^2$ ist $\| \left[\begin{smallmatrix}\alpha\\ \beta\end{smallmatrix}\right] \| = \sqrt{\alpha^2 + \beta^2}$.]

- Das Spielzeug bewegt sich immer in Richtung der Stange. Daher gilt für den Geschwindigkeitsvektor $v_S = [\dot{x}, \dot{y}]^T$ des Spielzeugs:

$$\lambda \begin{bmatrix} X - x \\ Y - y \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \text{ mit } \lambda \in \mathbb{R} \tag{1.2}$$

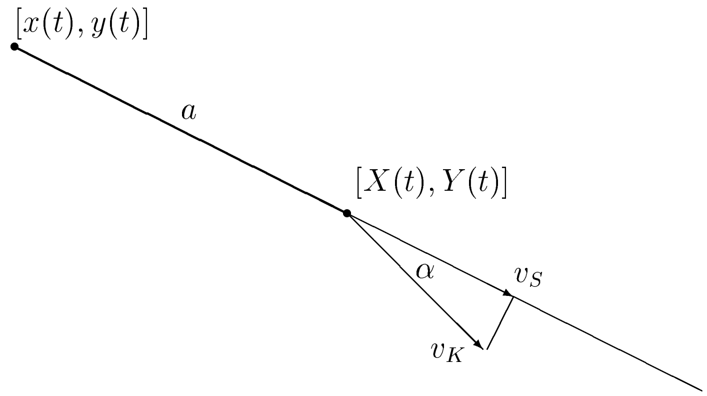


Abbildung 1.1: Kind mit Spielzeug

- Der Betrag der Geschwindigkeit des Spielzeugs sowie das Vorzeichen hängen von der Richtung des Geschwindigkeitsvektors $v_K = [\dot{X}, \dot{Y}]^T$ des Kindes ab. [Geht das Kind zum Beispiel im Kreis mit Radius a und ist das Spielzeug am Anfang im Kreiszentrum, so wird es sich überhaupt nicht bewegen. Geht das Kind in Stangenrichtung auf das Spielzeug zu, so ist v_S in negativer Stangenrichtung.]

Anhand von der Abbildung 1.1 sieht man leicht, daß $\|v_S\| = \|v_K\|\ |\cos\alpha|$ und das Vorzeichen von λ in (1.2) gerade dasjenige von $\cos\alpha$ sein muß.
Da für zwei Vektoren $r, s \in \mathbb{R}^n$ gilt $r^T s = \|r\|\ \|s\| \cos\beta$, wobei β den Winkel zwischen den beiden Vektoren bezeichnet, läßt sich v_S folgendermaßen berechnen. Wir normieren zunächst den Stangenrichtungsvektor auf Einheitslänge, d.h. wir berechnen

$$w = \frac{1}{\| \begin{bmatrix} X-x \\ Y-y \end{bmatrix} \|} \begin{bmatrix} X-x \\ Y-y \end{bmatrix} = \frac{1}{a} \begin{bmatrix} X-x \\ Y-y \end{bmatrix}. \tag{1.3}$$

Dann ist $v_K^T w = \|v_K\|\ \|w\|\ \cos\alpha$. Also kann man v_S berechnen als $v_S = |(v_K^T w) w|$. Wegen $a = \| \begin{bmatrix} X-x \\ Y-y \end{bmatrix} \|$ erhalten wir aus (1.2) nun $|\lambda| a = \|v_S\| = |v_K^T w|\ \|w\|$ und da λ und $v_K^T w$ dasselbe Vorzeichen haben, nämlich $\operatorname{sign}(\cos\alpha)$, und $\|w\| = 1$, so ist $\lambda a = (v_K^T w)$, bzw.

$$\lambda = \tfrac{v_K^T w}{a}.$$

Damit wird (1.2) zu

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{v_K^T w}{a} \begin{bmatrix} X-x \\ Y-y \end{bmatrix} = (v_K^T w) w, \tag{1.4}$$

wobei letzteres aus $a = \| \begin{bmatrix} X-x \\ Y-y \end{bmatrix} \|$ und (1.3) folgt.

Dies ist ein zweidimensionales gewöhnliches Differentialgleichungssystem erster Ordnung für die unbekannte (zweidimensionale) Funktion $\binom{x(t)}{y(t)}$. Gewöhnlich heißt das Differentialgleichungssystem weil nur nach einer Variablen, nämlich hier der Zeit t abgeleitet wird, im Gegensatz zu partiellen Differentialgleichungssystemen, bei denen Funktionen mehrerer Variabler mit ihren partiellen Ableitungen auftreten. Erster Ordnung heißt das System, weil nur erste Ableitungen, keine höheren, auftreten. Allgemein kann man so ein System schreiben als

$$\dot{z}(t) \;=\; f(t, z(t)),$$

wobei $z(t) = [z_1(t), \ldots, z_n(t)]^T$ ein Vektor von n Funktionen ist und $f(t, z(t))$ irgendeinen im allgemeinen nicht linearen Zusammenhang zwischen t und $z(t)$ beschreibt.
Matlab bietet zur numerischen Lösung einer gewöhnlichen Differentialgleichung (engl.: ordinary differential equation, abgekürzt: ODE) zwei m-files, ode23 und ode45 an. Mit dem Befehl 'help ode45' in Matlab erhält man die in Abbildung 1.2 abgedruckte Information über die Matlab-Funktion `ode45`, welche wir verwenden wollen.
Die in den Abbildungen 1.3 - 1.5 abgedruckten m-files berechnen eine Näherungslösung der Differentialgleichung (1.4) für den Fall, daß sich das Kind auf dem Kreis $X(t) = 5\cos(t)$; $Y(t) = 5\sin(t)$ in der Ebene bewegt und sich das Spielzeug zu Anfang in der Position $(10, 0)$ befindet, d.h. insbesondere startet das Kind bei $(X(0), Y(0)) = (5, 0)$ und es ist $a = 5$. Das m-file bsp1a.m benötigt die beiden m-file f.m und kind.m, welche den Wert der rechten Seite f(t,z) der Differentialgleichung, bzw. die Position des Kindes zur Zeit t berechnen.
Ruft man in Matlab nun 'bsp1a' auf, so erhält man die in Abbildung 1.6 wiedergegebene graphische Darstellung der Lösung. Der Weg des Kindes ist hierbei die gelbe Linie, der entsprechende Weg des Spielzeugs die grüne Linie. Bei einem anderen Weg des Kindes muß lediglich der m-file kind.m entsprechend ausgewechselt werden. Bei einer anderen Anfangsposition des Spielzeuges der Wert $y(0)$ im Hauptfile. Probieren wir dies noch aus für einen sinusförmigen Weg des Kindes, der Form $X(t) = t$ und $Y(t) = 5\sin(t)$ mit der Startposition $(0, 0)$ für das Kind und $(0, 10)$ für das Spielzeug (also $a = 10$) und für den Kreisweg des Kindes wie oben, aber mit der veränderten Startposition $(0, 10)$. Wir erhalten die in den Abbildungen 1.7 und 1.8 wiedergegebenen graphischen Darstellungen der Näherungslösungen.

```
ODE45  Solve differential equations, higher order method.
       ODE45 integrates a system of ordinary differential equations using
       4th and 5th order Runge-Kutta formulas.
       [T,Y] = ODE45('yprime', T0, Tfinal, Y0) integrates the system of
       ordinary differential equations described by the M-file YPRIME.M,
       over the interval T0 to Tfinal, with initial conditions Y0.
       [T, Y] = ODE45(F, T0, Tfinal, Y0, TOL, 1) uses tolerance TOL
       and displays status while the integration proceeds.

       INPUT:
       F     - String containing name of user-supplied problem description.
               Call: yprime = fun(t,y) where F = 'fun'.
               t      - Time (scalar).
               y      - Solution column-vector.
               yprime - Returned derivative column-vector; yprime(i) = dy(i)/dt.
       t0    - Initial value of t.
       tfinal- Final value of t.
       y0    - Initial value column-vector.
       tol   - The desired accuracy. (Default: tol = 1.e-6).
       trace - If nonzero, each step is printed. (Default: trace = 0).

       OUTPUT:
       T  - Returned integration time points (column-vector).
       Y  - Returned solution, one solution column-vector per tout-value.

       The result can be displayed by: plot(tout, yout).

       See also ODE23, ODEDEMO.
```

Abbildung 1.2: MATLAB's ode45

Auf dem Weg vom Ausgangsproblem zur berechneten Lösung haben wir hier einige Ungenauigkeit in Kauf genommen. Wir haben zunächst zur Beschreibung der praktischen Situation ein mathematisches Modell gebildet, nämlich die Differentialgleichung (1.4), die die zeitliche Entwicklung des Weges des Spielzeugs beschreibt. Dabei haben wir einige idealisierende Annahmen gemacht. Wir haben zum Beispiel wie üblich bei den Bewegungsgleichungen die Reibung vernachlässigt. Störeinflüsse durch mögliche Wegunebenheiten werden auch nicht berücksichtigt. Die Fehler, die man hierdurch macht, bezeichnet man als Modellfehler. Hierbei bedeutet 'Fehler' nicht, daß man etwas falsch gemacht hat, sondern daß man gewisse unvermeidbare Abweichungen vom "idealen", d.h. hier realitätsgenauen Ergebnis, erhält. In das mathematische Modell gehen Daten ein, wie hier die Angaben für den Startpunkt des Spielzeugs, die man unter Umständen nur mit einer beschränkten Genauigkeit messen kann oder die aus anderen Berechnungen mit einem gewissen Fehler stammen. Den hierdurch entstehenden Fehler nennt man Datenfehler. Anschließend ist

```
% bsp1a.m
%
% Kind mit Spielzeug:
%   - Kind laeuft im Kreis.
%   - Position des Spielzeugs konvergiert gegen Kreismittelpunkt.
%
y0 = [10 0]';
[t y] = ode45('f',[0 100],y0);
clf; hold on;
axis([-6 10 -6 10]);
axis('square');
[X, Xs, Y, Ys] = kind(t);
title ('Kind und Spielzeug.');
stickhandle = line ('Color', 'red', 'EraseMode', 'xor', ...
               'LineStyle', '-', 'XData', [], 'YData', []);
for k=1:size(y,1)-1,
  plot ([X(k), X(k+1)], [Y(k), Y(k+1)], '-', ...
        'Color', 'yellow', 'EraseMode', 'none');
  plot ([y(k,1), y(k+1,1)], [y(k,2), y(k+1,2)], '-', ...
        'Color', 'green', 'EraseMode', 'none');
  set (stickhandle, 'XData', [X(k+1), y(k+1,1)], ...
       'YData', [Y(k+1), y(k+1,2)]);
  drawnow;
  for k=1:1e4, end
end
hold off;
```

Abbildung 1.3: m-file bsp1a.m: Kind mit Spielzeug

```
function [X, Xs, Y, Ys] = kind(t);
%
% Berechnet die Position des Kindes zur Zeit t fuer kreisfoermigen Weg.
% (X,Y)   bezeichnet die Position des Kindes,
% (Xs,Ys) den Geschwindigkeitsvektor zum Zeitpunkt t.
%
% Aufruf:
% [X, Xs, Y, Ys] = kind(t)

X  =  5*cos(t);  Y  =  5*sin(t);
Xs = -5*sin(t);  Ys =  5*cos(t);
```

Abbildung 1.4: m-file kind.m zur Berechnung der Position des Kindes zur Zeit t

```
function zs = f(t,z)
%
% Berechnet den Wert der rechten Seite f(t,z) der Differentialgleichung
%
%    d/dt z = f(t,z) = (v'w) w
%
% zum Zeitpunkt t. Hier ist z = [x,y], v die Geschwindigkeit des Kindes
% zum Zeitpunkt t, und w = [X - x,Y - y]/norm([X - x,Y - y]), wobei
% (x,y) die Koordinaten des Spielzeugs und (X,Y) die Koordinaten des Kindes
% zum Zeitpunkt t bezeichnet.
%
% Benoetigt die Funktion 'kind'.

[X Xs Y Ys] = kind(t);
v =[Xs; Ys];
w =[X-z(1); Y-z(2)];
w = w/norm(w);
zs = (v'*w)*w;
```

Abbildung 1.5: m-file f.m zur Berechnung des Wertes der rechten Seite der DGL

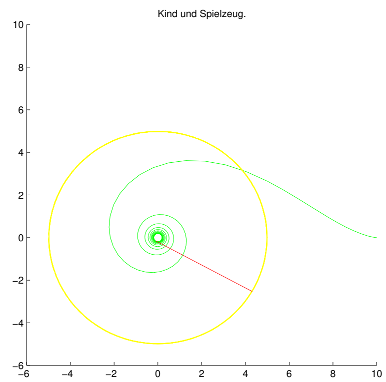


Abbildung 1.6: Kind mit Spielzeug: kreisförmiger Weg

das mathematische Problem des Modells zu Lösen. In einigen wenigen Fällen gelingt dies durch eine mathematische Analyse, die uns auf theoretischem Weg zu einer analytischen Lösung führt. In den weitaus meisten Fällen ist es erforderlich, eine numerische Methode zu benutzen, wie hier ein Verfahren zur näherungsweisen Lösung der gewöhnlichen Differentialgleichung. Dabei werden, wie wir später sehen werden, Näherungswerte für die Lösung der Differentialgleichung an bestimmten (sehr vielen) Punkten errechnet und nicht die exakte Lösungsfunktion. Den hierdurch entstehenden Fehler bezeichnet man als Verfahrensfehler. Letztendlich ist das numerische Verfahren in einen Algorithmus umzusetzen und auf einem

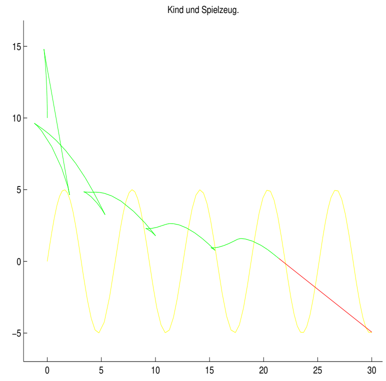


Abbildung 1.7: Kind mit Spielzeug: sinusförmiger Weg

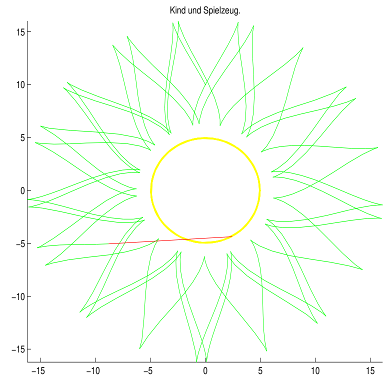


Abbildung 1.8: Kind mit Spielzeug: kreisförmiger Weg

geeigneten Rechner zu implementieren. Im allgemeinen rechnet man dabei mit numerischen Manipulationen in der Gleitpunktarithmetik des Rechners (im Gegensatz zu symbolischen Manipulation wie etwa mit MAPLE oder MACSYMA). Dabei entstehen in jedem Schritt Rundungsfehler, die sich im Ergebnis fortpflanzen und deren Einfluß auf den Fehler im Rechnerergebnis analysiert werden muß . All diese Fehler gehen in das vom Rechner schließlich gelieferte Ergebnis als Gesamtfehler ein und ergeben die Abweichung vom realen Ergebnis des Anwenderproblems.

Den Lösungsweg vom Anwenderproblem bis zum Ergebnis, das der Rechner als Näherungswert für die Lösung des Problems liefert, kann man grob in folgendem Schema darstellen.

Anwendungsproblem		Beispiel: Kind mit Spielzeug	**Fehler**
Modellbildung		Vernachlässigung von Bodenunebenheiten	**Modellfehler**
Math. Problem, Daten		Diff.gleichung, $(x(0), y(0))$	**Datenfehler**
Math. Analyse	Numer. Methode	Lsg. von Diffgl.	**Verfahrensfehler**
	Impl. und Analyse	Prog. ODE45, Fehleranalyse	**Rundungsfehler**
analyt. Lsg.	num. Lsg.	$(x(t_j),\ y(t_j))$ für viele t_j	
exakte Lsg.	Näherungslsg.		**Gesamtfehler**

Die numerische Mathematik beschäftigt sich überwiegend mit dem Weg vom mathematischen Problem zum Ergebnis des Algorithmus. Ein großes Gewicht liegt dabei auf der Entwicklung und Analyse des numerischen Verfahrens. Bei der Fehleranalyse und Effizienzbetrachtung spielt aber auch die spezielle Implementierung eine wichtige Rolle.

Kapitel 2

Fehleranalyse

2.1 Zahldarstellung

Die Darstellung eines REAL-Wertes x in einem Rechner ist

$$x = m \cdot B^e.$$

m heißt Mantisse, B Basis und e Exponent. Bei der Konstruktion des Rechners wird entschieden, wieviele Stellen (Bits) für die Mantisse und wieviele für den Exponenten reserviert werden. Und natürlich wird B festgelegt, z.B. $B = 2$ oder $B = 16$ oder bei Taschenrechnern $B = 10$. Dann reicht es, das Paar (m, e) zu speichern. Aus praktischen Gründen wünscht man, daß jeder Wert genau eine Darstellung hat. Wegen

$$x = m \cdot B^e = (m \cdot B) \cdot B^{e-1} = (m/B) \cdot B^{e+1}$$

gelangt man zu einer eindeutigen Darstellung, wenn man die folgende normalisierte Gleitpunktdarstellung fordert:

$$\begin{array}{ll} m = 0, e = 0 & \text{für } x = 0 \\ x = m \cdot B^e & \text{mit } 1/B \leq |m| < 1 \text{ für } \quad x \neq 0 \\ & \\ m = v_m \; m_1 \; m_2 \ldots m_p & p \text{ Stellen zur Basis } B,\; v_m \text{ Vorzeichen, } m_j \text{ Ziffer} \\ e = v_e \; e_1 \; e_2 \ldots e_q & q \text{ Stellen zur Basis } B,\; v_e \text{ Vorzeichen, } e_j \text{ Ziffer} \end{array}$$

$$m_1 m_2 \ldots m_p | e = (m_1 \cdot B^{-1} + m_2 \cdot B^{-2} + \ldots + m_p \cdot B^{-p}) \cdot B^e$$

Hiermit sind $2(B-1)B^{p-1}(2B^q - 1) + 1$ Zahlen darstellbar.
Die Menge aller darstellbaren Zahlen bzgl. der Basis B mit p Stellen für die Mantisse und q Stellen für den Exponenten sei mit $\mathbb{M}(B, p, q)$ bezeichnet. x_{max} bezeichne die größte, x_{min} die kleinste positive darstellbare Zahl.

Am Beispiel $B = 2,\ p = 3,\ q = 2$ soll die Darstellung verdeutlicht werden; die Darstellung ist also $v_m m_1 m_2 m_3 | v_e\ e_1 e_2$.
Die verfügbaren Mantissen sind

Basis 2		Basis 10
± 100	$\widehat{=}$	$\pm 1/2$
± 101	$\widehat{=}$	$\pm 5/8$
± 110	$\widehat{=}$	$\pm 3/4$
± 111	$\widehat{=}$	$\pm 7/8$

Die verfügbaren Exponenten sind

$$\begin{array}{lcl} \pm 00 & \widehat{=} & 0 \\ \pm 01 & \widehat{=} & \pm 1 \\ \pm 10 & \widehat{=} & \pm 2 \\ \pm 11 & \widehat{=} & \pm 3 \end{array}$$

Die darstellbaren Zahlen sind daher

$$\begin{array}{lcl}
\pm 000| \pm 00 & \widehat{=} & 0 \\[1ex]
\pm 100| \pm 00 & \widehat{=} & \pm\frac{1}{2} \cdot 2^0 = \pm\frac{1}{2} \\[1ex]
\pm 100| + 01 & \widehat{=} & \pm\frac{1}{2} \cdot 2^1 = \pm 1 \\[1ex]
\pm 100| - 01 & \widehat{=} & \pm\frac{1}{2} \cdot 2^{-1} = \pm\frac{1}{4} \\[1ex]
\pm 100| + 10 & \widehat{=} & \pm\frac{1}{2} \cdot 2^2 = \pm 2 \\[1ex]
\pm 100| - 10 & \widehat{=} & \pm\frac{1}{2} \cdot 2^{-2} = \pm\frac{1}{8} \\[1ex]
\pm 100| + 11 & \widehat{=} & \pm\frac{1}{2} \cdot 2^3 = \pm 4 \\[1ex]
\pm 100| - 11 & \widehat{=} & \pm\frac{1}{2} \cdot 2^{-3} = \pm\frac{1}{16} \\[1ex]
\pm 101| \pm 00 & \widehat{=} & \pm\frac{5}{8} \cdot 2^0 = \pm\frac{5}{8} \\[1ex]
\pm 101| + 01 & \widehat{=} & \pm\frac{5}{8} \cdot 2^1 = \pm\frac{5}{4} \\[1ex]
\pm 101| - 01 & \widehat{=} & \pm\frac{5}{8} \cdot 2^{-1} = \pm\frac{5}{16} \\
\vdots & & \\
\pm 111| + 11 & \widehat{=} & \pm\frac{7}{8} \cdot 2^3 = \pm 7 \\[1ex]
\pm 111| - 11 & \widehat{=} & \pm\frac{7}{8} \cdot 2^{-3} = \pm\frac{7}{64}
\end{array}$$

Es sind also insgesamt 57 Zahlen darstellbar, die kleinste darstellbare positive Zahl ist $\frac{1}{2} \cdot 2^{-3} = 2^{-4}$, die größte darstellbare positive ist $\frac{7}{8} \cdot 2^3 = 7$. In der Nähe von 0 liegen die Zahlen am dichtesten.
Die Zahl 0.1 z.B. ist hier nicht exakt darstellbar. Denn

$$0.1 \hat{=} +000110011001100\ldots| +00$$

und damit wird 0.1 dargestellt als

$$+110| -11$$

was der Dezimalzahl $\frac{3}{32} = 0.09375$ entspricht.
Allgemein gilt:
Jedes reelle x wird im Rechner als $\overline{x} \in \mathbb{M}(B,p,q)$ dargestellt mit $\overline{x} = x(1+\varepsilon)$. Genauer gilt für

$$\begin{aligned} x &= \pm(\textstyle\sum_{i=1}^{\infty} m_i B^{-i}) B^e \text{ mit } m_1 \neq 0 \text{ und } x \in [x_{min}, x_{max}] \\ \overline{x} &= \begin{cases} \pm(\sum_{i=1}^{p} m_i B^{-i}) B^e & \text{für } m_{p+1} < B/2 \\ \pm(\sum_{i=1}^{p} m_i B^{-i} + B^p) B^e & \text{für } m_{p+1} \geq B/2 \end{cases} \end{aligned}$$

Für $|x| < x_{min}$ gilt häufig $\overline{x} = 0$, $|x| > x_{max}$ ist meist nicht darstellbar.
Für den absoluten Fehler in der Zahldarstellung gilt

$$|\overline{x} - x| \leq \frac{1}{2} B^{-p} B^e,$$

für den relativen Fehler

$$|\frac{\overline{x} - x}{x}| < \frac{B}{2} B^{-p}.$$

Die Größe $\frac{B}{2} B^{-p} := \text{eps}$ bezeichnet man als Maschinengenauigkeit. Eine äquivalente Definition ist

$$\text{eps} := \min\{\delta : |\overline{(1+\delta)}| > 1\}.$$

Es ist also stets $\overline{x} = x(1+\varepsilon)$ mit $|\varepsilon| < \text{eps}$ für $|x| \in [x_{min}, x_{max}]$.
Bereits beim Einlesen/Speichern/Ausgaben reeller Daten können Fehler entstehen.

2.2 Gleitpunktoperationen

Als nächstes müssen arithmetische Operationen in $\mathbb{M}(B,p,q)$ erklärt werden. Legt man die üblichen Definitionen zugrunde, dann wird das Ergebnis im allgemeinen nicht wieder in $\mathbb{M}(B,p,q)$ liegen. Deshalb müssen neben der Simulation der reellen Zahlen auch die arithmetischen Operationen simuliert werden. Das geschieht in den Gleitpunktoperationen. So seien die Gleitpunktoperationen

$$\oplus, \ominus, \otimes, \oslash$$

durch die üblichen Definitionen festgelegt, mit anschließendem Runden des Ergebnisses, damit wieder ein Wert aus $\mathbb{M}(B,p,q)$ entsteht. Die Operationen sind nicht definiert, wenn das Ergebnis außerhalb des durch $\mathbb{M}(B,p,q)$ überdeckten Bereichs liegt.
Für die Addition zweier Zahlen $x, y \in \mathbb{R}$ ergibt sich der folgende Fehler, wobei für x und y gleiches Vorzeichen vorausgesetzt wird.

$$\begin{aligned}\overline{x} \oplus \overline{y} &= (\overline{x} + \overline{y})(1+\varepsilon_3) \\ &= (x(1+\varepsilon_1) + y(1+\varepsilon_2))(1+\varepsilon_3)\end{aligned}$$

mit $|\varepsilon_i| \leq \text{eps}$, also

$$\begin{aligned}|\overline{x} \oplus \overline{y}| &\leq |x+y|(1+\text{eps})^2 \\ &\approx |x+y|(1+2\text{eps}).\end{aligned}$$

Für die Multiplikation gilt entsprechend:

$$\begin{aligned}\overline{x} \otimes \overline{y} &= (\overline{x}\,\overline{y})(1+\varepsilon_3) \\ &= x(1+\varepsilon_1)y(1+\varepsilon_2)(1+\varepsilon_3) \\ |\overline{x} \otimes \overline{y}| &\leq |xy|(1+\text{eps})^3 \\ &\approx |xy|(1+3\ \text{eps}).\end{aligned}$$

Bei Addition und Multiplikation entstehen also nur recht kleine Fehler.
Bei der Subtraktion dagegen ergibt sich die folgende kritische Situation:

$$\begin{aligned}\overline{x} \ominus \overline{y} &= (\overline{x} - \overline{y})(1+\varepsilon_3) \\ &= (x - y + x\varepsilon_1 - y\varepsilon_2)(1+\varepsilon_3)\end{aligned}$$

Ein Abschätzung nach oben ist nicht möglich. Der relativer Rundungsfehler wird jetzt abhängig von der Größe der Operanden. So ergibt sich für die extreme Situation $x = y + \text{eps}$ die Abschätzung:

$$\begin{aligned}|\overline{x} \ominus \overline{y}| &\leq |x - y + y(\varepsilon_1 - \varepsilon_2)|(1+\text{eps}) \\ &\leq (|x-y| + 2|y|\text{eps})(1+\text{eps}) \\ &= |x-y|(1+2|y|)(1+\text{eps}) \\ &\approx |x-y|(1+2|y|)\end{aligned}$$

Vorsicht ist also geboten, wenn Differenzen fast gleichgroßer Zahlen auftreten, da die Stellenauslöschung zu sehr großen Rundungsfehlern führt. Dazu zwei Beispiele:
1) In $\mathbb{M}(10,8,q)$ ist $\text{eps} = 5 \cdot 10^{-8}$. Sei

$$\begin{aligned}a &= 1000000.123 \\ b &= 1000000.012\end{aligned}$$

Dann folgt

$$\begin{aligned}\overline{a} &= 1000000.1\\ \overline{b} &= 1000000.0\end{aligned}$$

und $\overline{a} \ominus \overline{b} = 0.1$.
Für die Differenz $a - b$ ist in $\mathbb{M}(10, 8, q)$ bereits die 2. Stelle nach dem Komma unsicher, d.h. es muß mit einem Rundungsfehler von $5 \cdot 10^{-1}$ gerechnet werden.
2) Summation von Potenzreihen, deren Glieder alternierende Vorzeichen haben.
Zur Berechnung von e^x wird von der Reihenentwicklung

$$e^x = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots$$

ausgegangen, nach der in $\mathbb{M}(10, 5, q)$ $e^{-5,5}$ berechnet werden soll, also

$$\begin{array}{lcr} e^{-5,5} & = & 1.000 \\ & - & 5.500 \\ & + & 15.125 \\ & - & 27.730 \\ & + & 38.129 \\ & - & 41.942 \\ & + & 38.446 \\ & - & 30.208 \\ & \ldots & \\ \hline & + & 0.0026363 \end{array}$$

Bei einer Rechengenauigkeit von 5 Dezimalstellen ist bereits die 4. Stelle nach dem Dezimalpunkt, also die zweite Stelle im Ergebnis, unsicher. Dem überlagert sich die Fortpflanzung der Rundungsfehler bei den Gleitpunktoperationen, was hier das Ergebnis völlig unbrauchbar macht. Der exakte Wert lautet auf 5 Stellen genau:

$$e^{-5,5} = 0.0040868.$$

Die Stellenauslöschung wäre vermieden worden, wenn man statt $e^{-5,5}$ direkt $e^{5,5}$ nach der Reihenentwicklung ermittelt und dann den Kehrwert gebildet hätte:

$$1/(1 + 5.5 + 15.125 + \ldots) = 0.0040865.$$

Neben der fatalen Auswirkung bei der Differenz fast gleicher Zahlen ergeben sich kritische Situationen, wenn bei der Programmierung nicht berücksichtigt wird, daß für Gleitpunktaddition und -multiplikation weder das Assoziativ- noch das Distributivgesetz gilt:
Sei $u, v, w \in \mathbb{M}(B, p, q)$, dann ist im allgemeinen

$$\begin{aligned}(u \oplus v) \oplus w &\neq u \oplus (v \oplus w)\\ (u \otimes v) \otimes w &\neq u \otimes (v \otimes w)\\ u \otimes (v \oplus w) &\neq (u \otimes v) \oplus (u \otimes w).\end{aligned}$$

Ein Beispiel dazu ist in $\mathbb{M}(10,5,1)$:

$$\left.\begin{array}{rcl} a=\overline{a} & = & 0.98765 \\ b=\overline{b} & = & 0.012424 \\ c=\overline{c} & = & -0.0065432 \end{array}\right\} a+b+c=0.9935308$$

aber

$$\begin{array}{lcl} (a\oplus b) & = & 1.0001=(0.98765\oplus 0.01242) \\ (b\oplus c) & = & 0.0058808=(0.012424\oplus -0.006543) \\ (a\oplus b)\oplus c & = & 0.99356=(1.0001\oplus -0.0065) \\ a\oplus(b\oplus c) & = & 0.99353=(0.98765\oplus 0.005881) \end{array}$$

D.h. $a\oplus(b\oplus c)$ liefert das richtige Ergebnis, nicht aber $(a\oplus b)\oplus c$.
Aus den besprochenen Schwierigkeiten mit der Gleitpunktarithmetik und der Zahldarstellung folgt, daß es unsinnig ist, zwei Gleitpunktzahlen auf Gleichheit zu überprüfen. Ihre Berechnung wird im allgemeinen über verschiedene Algorithmen erfolgen und damit zu unterschiedlichen Rundungsfehlern führen. Vielmehr muß die Abfrage als $|x-y|<\delta$ mit geeignetem δ gefaßt werden.

2.3 Fehleranalyse

Zur Beurteilung darüber, wie gut sich ein numerisches Verfahren zur Lösung eines vorliegenden Problems für die Durchführung auf einem Rechner eignet, wird man im wesentlichen folgende Kriterien berücksichtigen:

1. Genauigkeit des Ergebnisses, Verläßlichkeit

2. Rechenzeitbedarf, Speicherbedarf.

Die Bewertung der Eigenschaften 2) hängt stark von der äußeren Konfiguration des zur Verfügung stehenden Rechners ab. Die Beurteilung eines Verfahrens oder der Vergleich zweier Verfahren ist i.a. gut durchführbar.
Die meisten numerischen Verfahren arbeiten nur richtig, falls die gegebenen Daten des Problems gewissen Bedingungen genügen. Bei den speziell vorliegenden Daten eines Problems ist häufig nicht ersichtlich oder leicht überprüfbar, ob die für das betrachtete Verfahren notwendigen Voraussetzungen erfüllt sind. Dies gilt insbesondere, wenn es sich hierbei um ein während der Rechnung zu erzeugendes Zwischenergebnis in einem komplexeren Verfahren handelt. Die Verläßlichkeit eines Verfahrens beurteilt man danach, wie ein solches Versagen des Verfahrens den Rechenprozeß beeinflußt, z.B. durch Abbruch, Meldung oder Korrektur. Insbesondere sollte es nicht möglich sein, daß aufgrund eines Versagens des Verfahrens oder Teilen des Verfahrens ein sinnvoll erscheinendes, aber völlig falsches Ergebnis produziert wird.
Um Aussagen über die Genauigkeit des Ergebnisses zu erhalten, ist eine detaillierte Betrachtung des numerischen Verfahrens bezüglich aller möglichen auftretenden Fehler und

ihrer Einflüsse auf das Ergebnis erforderlich. In diesem Zusammenhang versteht man unter Fehler jeweils die Abweichung eines nach bestimmten Vorschriften produzierten Ergebnisses vom gewünschten Resultat, also zum Beispiel nicht einen Programmfehler. Man unterscheidet hier drei wesentliche Fehlerarten: 1. Datenfehlereffekte, 2. Verfahrensfehler, 3. Rundungsfehler.

2.3.1 Datenfehlereffekte

Bei der Gewinnung eines mathematischen Problems aus einem Anwenderproblem werden fast stets eine Reihe idealisierender Annahmen gemacht, etwa indem Größen von geringem Einfluß vernachlässigt und funktionale Zusammenhänge stark vereinfacht werden. Schon aus diesem Grund wird die Lösung des mathematischen Problems von der Lösung des Ausgangsproblems aus der Anwendung abweichen. Zudem erhält man die Daten des mathematischen Problems, zum Beispiel die Elemente einer Matrix, deren Eigenwerte gesucht sind, häufig als Ergebnisse von Messungen, die nur mit beschränkter Genauigkeit durchgeführt werden können, oder als bereits mit Fehlern behaftete Ergebnisse anderer Berechnungen. Faßt man das Lösen des Problems als Auswerten der Funktion $f : D \to W$ für einen gegebenen Datenwert aus D auf, so hat man also statt des exakten Datenwertes x nur den verfälschten Datenwert $\tilde{x}$ zur Verfügung. Mit den Daten ändert sich natürlich auch die Lösung $f(x)$ des Problems zu $f(\tilde{x})$, Diese durch fehlerhafte Daten entstehende Abweichung in der Lösung, $f(x) - f(\tilde{x})$, bezeichnen wir als *Datenfehlereffekt.* Die Empfindlichkeit der Lösung des mathematischen Problems bezüglich (kleiner) Änderungen in den Daten nennt man *Kondition des Problems.* Das Problem wird *gut konditioniert* genannt, wenn kleine Änderungen des Datenwertes x auch nur kleine Änderungen in $f(x)$ bewirken, andernfalls bezeichnet man es als *schlecht konditioniert.* Die gute oder schlechte Kondition ist also eine Eigenschaft des mathematischen Problems und *nicht* des gewählten numerischen Verfahrens zur Berechnung seiner Lösung.

Ein Beispiel dazu: Gesucht ist die kleinere Lösung von $x^2 + 2ax - b = 0$. Die exakte Lösung ist gegeben durch $x = -a + \sqrt{a^2 + b} = p(a, b)$. Sind $\tilde{a} = a + \varepsilon_1, \tilde{b} = b + \varepsilon_2$ die verfälschten Daten und $\tilde{x} = p(\tilde{a}, \tilde{b})$ die Lösung bzgl. $\tilde{a}, \tilde{b}$. Es folgt (Taylor-Formel)

$$p(\tilde{a}, \tilde{b}) - p(a, b) = \frac{\partial p}{\partial a}(a, b)\varepsilon_1 + \frac{\partial p}{\partial b}(a, b)\varepsilon_2 + \text{Terme höherer Ordnung}$$

und daher

$$\frac{\tilde{x} - x}{x} = \left(\frac{\partial p}{\partial a}(a, b)\frac{a}{x}\right)\frac{\varepsilon_1}{a} + \left(\frac{\partial p}{\partial b}(a, b)\frac{b}{x}\right)\frac{\varepsilon_2}{b} + \text{Terme höherer Ordnung}\ldots$$

Nun kann man für die Verstärkungsfaktoren der relativen Fehler $\frac{\varepsilon_1}{a}, \frac{\varepsilon_2}{b}$ zeigen:

$$\begin{aligned}
\tfrac{\partial p}{\partial a}(a,b) &= -1 + \tfrac{a}{\sqrt{a^2+b}} = \tfrac{-x}{\sqrt{a^2+b}} \\
\tfrac{\partial p}{\partial b}(a,b) &= \tfrac{1}{2\sqrt{a^2+b}} \\
\tfrac{\partial p}{\partial a}(a,b)\tfrac{a}{x} &= \tfrac{-a}{\sqrt{a^2+b}} \\
\tfrac{\partial p}{\partial b}(a,b)\tfrac{b}{x} &= \tfrac{a+\sqrt{a^2+b}}{2\sqrt{a^2+b}}
\end{aligned}$$

Wegen

$$|\frac{a}{\sqrt{a^2+b}}| \leq 1, \; |\frac{a+\sqrt{a^2+b}}{2\sqrt{a^2+b}}| \leq 1 \text{ für } b > 0$$

ist das Problem gut konditioniert, falls $b > 0$ und schlecht konditioniert, falls etwa $b \approx -a^2$.
D.h. unter der Voraussetzung $0 < b \ll 1 < a$ ist das Problem gut konditioniert.
Zur eigentlichen Berechnung der Lösung kann man die Formel $x = -a + \sqrt{a^2+b}$ oder $x = \frac{b}{a+\sqrt{a^2+b}}$ verwenden:
Algorithmus 1: $x = -a + \sqrt{a^2+b}$

$$\begin{aligned}
p &:= a * a \\
d &:= p + b \\
w &:= \sqrt{d} \\
x &:= -a + w
\end{aligned}$$

Algorithmus 2: $x = \frac{b}{a+\sqrt{a^2+b}}$

$$\begin{aligned}
p &:= a * a \\
d &:= p + b \\
w &:= \sqrt{d} \\
s &:= a + w \\
x &:= b/s
\end{aligned}$$

Für $a = 6.0002,\ b = 0.01$ und $\mathbb{M}(10,5,1)$ ergibt sich
nach Algorithmus 1 $x = 0.0007$
nach Algorithmus 2 $x = 0.00080122$.
Die exakte Lösung lautet $x = 0.0008333$.
In Algorithmus 1 tritt bei $x := -a + w$ Auslöschung auf, für den relativen Fehler $\frac{\sqrt{a^2+b}}{x}$eps gilt

$$|\frac{\sqrt{a^2+b}}{x}| \leq k \cdot \text{eps} \;\text{ mit } k > \frac{2a^2}{b} > 0.$$

Für $a \gg b$ kann der Verstärkungsfaktor k sehr groß sein, daher ist der Algorithmus 1 nicht gutartig. Algorithmus 2 hingegen ist gutartig.
Wir haben gesehen, daß die gute oder schlechte Kondition eines Problems eine Eigenschaft des Problems und nicht des numerischen Verfahrens zur Berechnung seiner Lösung ist. Zur Lösung gut konditionierter Probleme kann es gut- und schlechtartige numerische Verfahren geben.

2.3.2 Verfahrensfehler

Für einige mathematische Probleme gibt es numerische Verfahren, die nach einer gewissen Anzahl von Berechnungsschritten (bei exakter Rechnung) die gesuchte Lösung ermittelt haben. Das Gaußsche Eliminationsverfahren zur Lösung von linearen Gleichungssystemen ist zum Beispiel ein solches Verfahren. Für die meisten Aufgaben muß man jedoch iterative Verfahren verwenden, mit denen man schrittweise eine Folge von Näherungslösungen bestimmen kann, die gegen die gesuchte Lösung konvergiert. Dazu gehören zum Beispiel bis auf sehr spezielle Fälle alle Eigenwertbestimmungen von Matrizen, da die Eigenwerte als Nullstellen des charakteristischen Polynoms im allgemeinen nicht mit endlich vielen rationalen Operationen berechenbar sind. Bei solchen Verfahren bricht man, abhängig von geeigneten Kriterien, die Berechnung nach einer gewissen Anzahl von Schritten mit einer Näherungslösung $\tilde{y}$ ab. Die Abweichung $y - \tilde{y}$ dieser Näherungslösung von der gesuchten Lösung y des Problems nennt man *Verfahrensfehler* oder auch *Abbrechfehler*.

2.3.3 Rundungsfehler

Die Rundungsfehlerproblematik wurde in den Abschnitten Zahldarstellung und Gleitpunktoperationen ausführlich behandelt.

2.3.4 Vorwärts-/Rückwärtsanalyse

Bei der Durchführung einer Fehleranalyse für ein Verfahren gibt es unterschiedliche Vorgehensweisen. Bei der *Vorwärtsanalyse* verfolgt man sämtliche Fehler von Rechenschritt zu Rechenschritt und schätzt sofort für jedes Zwischenergebnis den bisher aufgetretenen akkumulierten Fehler ab. Für das Endergebnis steht dann eine Abschätzung des Gesamtfehlers auch direkt zur Verfügung. Die Verfolgung der Fehler von Schritt zu Schritt des Verfahrens in der *Rückwärtsanalyse* geschieht so, daß jedes Zwischenergebnis, das der Algorithmus liefert, als exakt berechneter Wert für gestörte Anfangsdaten interpretiert wird, d.h. der akkumulierte Fehler im Zwischenergebnis wird als Datenfehlereffekt gedeutet. Über den Gesamtfehler im Endergebnis erhält man so zunächst ebenfalls nur die Aussage, daß er einem Datenfehlereffekt aufgrund von Datenfehlern bestimmter Größe entspricht; diese bezeichnet man als *äquivalente Datenfehler*. Bei zwei verschiedenen Algorithmen für dasselbe Problem zeigt die Größe der äquivalenten Datenfehler dann an, welcher günstigere numerische Eigenschaften besitzt. Ergibt eine Rückwärtsanalyse, daß die äquivalenten Datenfehler in einem Algorithmus nicht viel größer sind als die im allgemeinen schon durch

die Rundung der gegebenen Daten entstehenden Darstellungsfehler, so besitzt ein solcher Algorithmus offensichtlich sehr gute numerische Eigenschaften; er heißt dann *numerisch stabil.* Um nun den Gesamtfehler abschätzen zu können, benötigt man Aussagen über die Auswirkungen der Störungen in den Anfangsdaten auf die Genauigkeit des Ergebnisses. Für viele Probleme gibt es dazu Abschätzungen, sogenannte *Störungssätze.* Erst die Kombination der Rückwärtsanalyse mit einem Störungssatz liefert also eine Abschätzung für den Fehler im Ergebnis, während man eine solche bei der Vorwärtsanalyse direkt erhält. Dennoch erweist sich die Kombination aus Rückwärtsanalyse und Störungssatz in vielen Fällen als vorteilhaft, da nur bei der Rückwärtsanalyse die Rechenarithmetik zu berücksichtigen ist, während bei der Herleitung eines Störungssatzes die Eigenschaften der reellen Zahlen wie Kommutativität und Assoziativität ausgenutzt werden können. Häufig erkennt man bei diesem Vorgehen auch, daß der durch die fast immer auftretenden Darstellungsfehler entstehende unvermeidbare Fehler, der durch einen Störungssatz manchmal gut geschätzt werden kann, schon recht groß ist und Ursache für einen großen Gesamtfehler ist. Bei einer Vorwärtsanalyse kann man dies im allgemeinen nicht so leicht erkennen, und man könnte dann dem Trugschluß erliegen, daß ein großer Fehler im Ergebnis stets das Resultat von Mängeln des Algorithmus ist.
Als ein Beispiel zur Vorwärts- und Rückwärtsanalyse wird nun der Algorithmus zur Berechnung von $x = a - \sqrt{a^2 - b}$ betrachtet:

$$\begin{array}{rcl} p & := & a * a \\ d & := & p - b \\ w & := & \sqrt{d} \\ x & := & a - w \end{array}$$

Bei der Fehleranalyse erhält man zunächst

$$\begin{array}{rcl} \tilde{p} & = & a^2(1+\varepsilon_1) \\ \tilde{d} & = & (\tilde{p} - b)(1+\varepsilon_2) \\ \tilde{w} & = & \sqrt{\tilde{d}}(1+\varepsilon_3) \\ \tilde{x} & = & (a - \tilde{w})(1+\varepsilon_4) \end{array}$$

mit $|\varepsilon_i| < \text{eps}$ für $i = 1, 2, 3, 4$, d.h.

$$\tilde{x} = [a - \sqrt{(a^2(1+\varepsilon_1) - b)(1+\varepsilon_2)}(1+\varepsilon_3))](1+\varepsilon_4).$$

Bei der Rückwärtsanalyse wird das Ergebnis als exakt berechneter Wert für gestörte Anfangsdaten interpretiert, d.h. der Ausdruck für $\tilde{x}$ ist so umzuformen, daß man eine Aussage der folgenden Art treffen kann: Die vom Rechner gelieferte Zahl $\tilde{x}$ ist die Lösung von

$$x^2 - 2a(1+\delta_1)x + b(1+\delta_2) = 0.$$

Der Ausdruck für $\tilde{x}$ muß daher in die Form

$$\tilde{x} = a(1+\delta_1) - \sqrt{(a(1+\delta_1))^2 - b(1+\delta_2)}$$

umgeformt werden.
Es gilt

$$\begin{aligned}\tilde{x} &= a(1+\varepsilon_4) - \sqrt{a^2(1+\varepsilon_1)(1+\varepsilon_2)(1+\varepsilon_3)^2(1+\varepsilon_4)^2 - b(1+\varepsilon_2)(1+\varepsilon_3)^2(1+\varepsilon_4)^2} \\ &= a(1+\varepsilon_4) - \sqrt{a^2(1+\varepsilon_4)^2(1+\varepsilon_5) - b(1+\varepsilon_6)}\end{aligned}$$

mit $1+\varepsilon_5 = (1+\varepsilon_1)(1+\varepsilon_2)(1+\varepsilon_3)^2$ und $1+\varepsilon_6 = (1+\varepsilon_2)(1+\varepsilon_3)^2(1+\varepsilon_4)^2$, so daß gilt $|\varepsilon_5| < 4$ eps, $|\varepsilon_6| < 5$ eps. Weiterhin folgt

$$\begin{aligned}\tilde{x} &= a(1+\varepsilon_4) - \sqrt{a^2(1+\varepsilon_4)^2 - b(1+\varepsilon_6) + a^2(1+\varepsilon_4)^2\varepsilon_5} \\ &= a(1+\varepsilon_4) - \sqrt{a^2(1+\varepsilon_4)^2 - b(1+\varepsilon_6 + \tfrac{a^2}{b}(1+\varepsilon_4)^2\varepsilon_5)} \\ &= a(1+\varepsilon_4) - \sqrt{(a(1+\varepsilon_4))^2 - b(1+\varepsilon_7)}\end{aligned}$$

mit $|\varepsilon_7| < (5 + \frac{6a^2}{b})$ eps. Die vom Rechner gelieferte Zahl $\tilde{x}$ ist daher die Lösung von

$$x^2 - 2a(1+\varepsilon_4)x + b(1+\varepsilon_7) = 0.$$

Bei der Vorwärtsanalyse ist das Ziel, eine Aussage über den relativen Fehler $|\frac{\tilde{x}-x}{x}|$ zu treffen. Daher wird hier versucht, $\tilde{x}$ so umzuformen, daß $\tilde{x} = x+$ Rest gilt. Unter Verwendung der Taylorformel für $\sqrt{d+h} = \sqrt{d} + \frac{1}{2\sqrt{d}}h+$ Terme höherer Ordnung ergibt sich

$$\begin{aligned}\tilde{x} &= [a - \sqrt{(a^2(1+\varepsilon_1) - b)(1+\varepsilon_2)}(1+\varepsilon_3)](1+\varepsilon_4) \\ &= a + a\varepsilon_4 - \sqrt{a^2(1+\varepsilon_1)(1+\varepsilon_2)(1+\varepsilon_3)^2(1+\varepsilon_4)^2 - b(1+\varepsilon_2)(1+\varepsilon_3)^2(1+\varepsilon_4)^2(1+\varepsilon_4)^2} \\ &= a + a\varepsilon_4 - \sqrt{a^2(1+\varepsilon_8) - b(1+\varepsilon_9)} \\ &= a - \sqrt{a^2 - b + (a^2\varepsilon_8 - b\varepsilon_9)} + a\varepsilon_4 \\ &= a - \sqrt{a^2 - b} - \tfrac{a^2\varepsilon_8 - b\varepsilon_9}{2\sqrt{a^2-b}} + a\varepsilon_4\end{aligned}$$

mit $|\varepsilon_8| \leq 6$ eps, $|\varepsilon_9| \leq 5$ eps und unter Vernachlässigung der Terme höherer Ordnung. Es folgt

$$\begin{aligned}|\tfrac{\tilde{x}-x}{x}| &= |\tfrac{a^2\varepsilon_8 - b\varepsilon_9 - 2a\varepsilon_4\sqrt{a^2-b}}{2\sqrt{a^2-b}(a-\sqrt{a^2-b})}| \\ &\leq 6 \text{ eps } |\tfrac{a^2-b-2a\sqrt{a^2-b}}{2\sqrt{a^2-b}(a-\sqrt{a^2-b})}|\end{aligned}$$

Kapitel 3

Numerische Basisverfahren

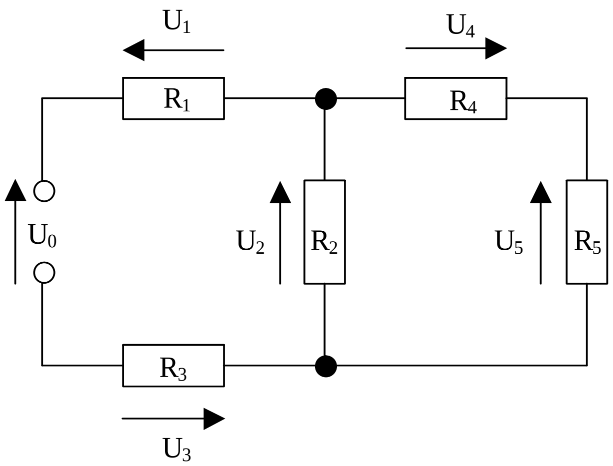
3.1 Lineare Gleichungssysteme

3.1.1 Gaußsche Eliminationsmethode

Wir beginnen mit dem klassischen Verfahren zur Auflösung eines linearen Gleichungssystems, der Gaußschens Eliminationsmethode. Sie basiert auf Überlegungen, die Carl Friedrich Gauß (1777 - 1855) in seinen 'Disquisitiones arithmeticae' 1801 veröffentlicht hat. Er berechnet mit den dort beschriebenen Methoden die Bahn des Planetoiden Ceres, dessen Wiederauffindung im gleichen Jahr an der von ihm vorausberechneten Stelle dann auch tatsächlich gelang (was ihm einigen Ruhm einbrachte).

Wir beginnen mit einem etwas einfacheren Beispiel:

Beispiel 3.1 *Wir betrachten das folgende elektrische Netzwerk (dieses Beispiel ist dem Buch Numerische Verfahren für Naturwissenschaftler und Ingenieure von H. Schwetlick und H. Kretzschmar, Fachbuchverlag Leipzig, 1991 entnommen)*



Bei vorgegebener Eingangsspannung U_0 sind die an den Widerständen $R_1, \ldots, R_5$ abfallenden Spannungen $U_1, \ldots, U_5$ zu berechnen. Zur Aufstellung der benötigten 5 Gleichungen *werden die Kichhoffschen Gesetze herangezogen: In jeder der beiden Maschen muß die Summe der Spannungen Null sein, was auf*

$$\begin{aligned} U_1 + U_2 + U_3 &= U_0 \\ U_2 + U_4 + U_5 &= 0 \end{aligned}$$

führt. In jedem der beiden Knoten muß die Summe der Ströme Null sein, d.h., es muß

$$\begin{aligned} -I_1 + I_2 - I_4 &= 0 \\ -I_2 + I_3 + I_5 &= 0 \end{aligned}$$

gelten, und durch R_4 und R_5 fließt derselbe Strom, d.h.

$$I_4 - I_5 = 0.$$

Dabei bezeichnet I_j den durch R_j fließenden Strom. Werden die Ströme nach dem Ohmschen Gesetz gemäß $I_j = \frac{U_j}{R_j} = U_j G_j$ durch die Spannung U_j und die Leitwerte $G_j = \frac{1}{R_j}$ ausgedrückt, ergibt sich das lineaere Gleichungssystem

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ -G_1 & G_2 & 0 & -G_4 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & -G_2 & G_3 & 0 & G_5 \\ 0 & 0 & 0 & G_4 & -G_5 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{bmatrix} = \begin{bmatrix} U_0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Etwas verallgemeinert ergibt sich aus diesem Beispiel eine Aufgabenstellung, die das Lösen eines *linearen Gleichungssystems* verlangt, nämlich

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = & b_1, \\ a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & = & b_2, \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{mn}x_n & = & b_m \end{array} \tag{3.1}$$

mit gegebenen Zahlen a_{ij} und b_i und gesuchten Zahlen $x_j, i = 1, 2, \dots, m,\ j = 1, 2, \dots, n$. *Die Koeffizienten a_{ij}, die rechten Seiten b_i* und *die Lösungen x_j* fassen wir in je einem rechteckigen Zahlenfeld (auch Matrix genannt) zusammen:

$$A := \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}, x := \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, b := \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

und schreiben für (3.1)

$$\begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

oder kurz:

$$Ax = b\,. \tag{3.2}$$

Wir gehen hier davon aus, daß $m = n$ ist. $A \in \mathbb{R}^{n\times n}$ ist also eine quadratische Matrix. Bevor wir mit der Berechnung der Lösung x beginnen, stellen wir uns die Frage: Wann ist das lineare Gleichungssystem (3.2) überhaupt lösbar? Aus den Mathematikvorlesungen kennen wir das folgende Resultat, das die Lösbarkeit quadratischer Gleichungssysteme mit Hilfe der Determinante der Matrix A charakterisiert.

Satz 3.2 *Sei $A \in \mathbb{R}^{n\times n}$ eine reelle quadratische Matrix mit $\det A \neq 0$ und $b \in \mathbb{R}^n$. Dann existiert genau ein $x \in \mathbb{R}^n$, so daß $Ax = b$.*

Falls $\det A \neq 0$, so läßt sich also die Lösung $x = A^{-1}b$ berechnen.

Ist die Dimension n (=Anzahl der Unbekannten) zu groß , so daß sich A nicht mehr vernünftig auf dem benutzten Rechner als zweidimensionales Feld bearbeiten läßt, so hat man für die numerische Behandlung ein qualitativ anderes Problem als im Fall moderater Dimensionen. Solche großen linearen Gleichungssysteme treten z.B. bei der Lösung partieller Differentialgleichungen auf. In diesem Abschnitt betrachten wir Verfahren für kleine bis mittelgroße Dimensionen.

In MATLAB gibt es eine sehr einfache Möglichkeit, die Lösung eines solchen Systems zu berechnen. Man gibt die Matrix A und den Vektor b ein und gibt ein: $x = A\backslash b$. Man erhält als Ausgabe den Lösungsvektor von $Ax = b$. Betrachten wir z.B. die Daten

$$A = \begin{bmatrix} 0.7 & -0.2 & -0.1 \\ -0.1 & 0.6 & -0.2 \\ -0.1 & -0.1 & 0.9 \end{bmatrix}, \qquad b = \begin{bmatrix} 20 \\ 40 \\ 0 \end{bmatrix}.$$

$x = A\backslash b$ liefert dann (auf 6 Stellen) die Lösung

$$x = \begin{bmatrix} 53.7313 \\ 80.5970 \\ 14.9254 \end{bmatrix}.$$

Dies ist eine gute numerische Lösung des Problems. Berechnen wir einmal das *Residuum*,

$$r = A * x - b,$$

so ist $\|r\|_2 = 5.02 * 10^{-15}$. (Können Sie erklären, warum $\|r\|_2 = 0$ nicht zu erwarten ist?)

Betrachten wir noch ein weiteres Beispiel, nämlich die sogenannte Hilbertmatrix der Dimension $n:\ H = (h_{ij})_{i,j\in 1,\ldots,n}$ gegeben duch

$$h_{ij} = \frac{1}{i+j-1} \qquad \text{für } i,j = 1,\ldots,n.$$

Wenn wir den Vektor b so wählen, daß die j-te Komponente gerade die j-te Zeilensumme von H ist, also

$$b_j = \sum_{i=1}^{n} h_{ij} \quad \text{für } j = 1, \dots, n, \tag{3.3}$$

so muß ja für den Vektor $y = [1, 1, \dots, 1]^T$ gelten

$$Hy = b. \tag{3.4}$$

Betrachten wir $n = 20$ und lösen das Gleichungssystem einmal wie eben. In MATLAB kann man dies so tun:

```
H= hilb(20); % Erzeugt die Hilbertmatrix der Dimension 20
b= sum(H)' ; %  Erzeugt $b$
x=H\b
```

Das errechnete Ergebnis x hat nicht viel mit dem Vektor $y = [1, 1, \dots, 1]^T$ gemein, z.B. ist $x_9 = 13.4100$, $x_{10} = -12.0762$, $x_{18} = -20.1033$, $x_{19} = 12.8201$. Hier scheint etwas nicht in Ordnung zu sein. Liegt es am numerischen Verfahren, das benutzt wird, oder gibt es andere Gründe für dieses katastrophale Ergebnis? Berechnen wir noch die Norm des Residuums, so finden wir

$$\|Hx - b\|_2 = 2.2232 \cdot 10^{-15}$$

Das Residuum für das "falsche" x ist also recht klein, obwohl die Lösung weit von der korrekten Lösung entfernt ist

$$\|y - x\| = 82.4524.$$

Offensichtlich kann man es auch bei diesem sehr einfachen Problem nicht dabei bewenden lassen, daß es gute Software zur Lösung gibt, die man dann eben, wenn es nötig ist, einfach benutzt. Sondern man muß sich die Aufgabe selbst und die verwendeten numerische Verfahren schon einmal genauer ansehen, um zum Beispiel das obige scheinbare Versagen des Verfahrens zu verstehen. Das Standard-Verfahren zur Lösung linearer Gleichungssysteme der Form $Ax = b$ mit $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, A nicht singulär, welches MATLAB bei Aufruf von $A\backslash b$ verwendet, ist das **Gaußsche Eliminationsverfahren** und seine Varianten. Mit diesem Verfahren führt man das Gleichungssystem durch elementare Operationen, die die Lösung x nicht verändern, über in eines, für das man die Lösung sehr leicht ausrechnen kann.

Am einfachsten ist sicherlich der Fall einer diagonalen Matrix A, bei dem das Gleichungssystem aus n unabhängigen skalaren Gleichungen besteht. Es macht ebenfalls keine Schwierigkeiten, ein quadratisches lineares Gleichungssystem zu lösen, wenn es folgende Gestalt (genannt *gestaffeltes Gleichungssystem*) hat:

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

in Matrix-Vektor-Notation kurz

$$Rx = z\,,$$

wobei R eine obere Dreiecksmatrix ist, d.h. $r_{ij} = 0$ für alle $i > j$. Offenbar erhalten wir x durch rekursive Auflösung, beginnend mit der Zeile n:

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

Diese Vorgehensweise nennt man *Rückwärtseinsetzen*.

Nun gilt für die obere Dreiecksmatrix R, daß $\det R = r_{11} \cdots r_{nn}$ und daher

$$\det R \neq 0 \iff r_{ii} \neq 0 \quad \text{für alle} \quad i = 1, \dots, n\,.$$

Der angegebene Algorithmus ist also genau dann durchführbar, wenn $\det R \neq 0$, also unter der Bedingung des Existenz- und Eindeutigkeitssatzes 3.2.

Von einem guten Rechenverfahren erwarten wir sicherlich, daß es die gestellte Aufgabe mit möglichst geringem Aufwand (an Rechenoperationen) löst.
Für den Rechenaufwand ergibt sich hier

a) für die i-te Zeile: je $n - i$ Additionen und Multiplikationen, eine Division

b) insgesamt für die Zeilen n bis 1:

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

Multiplikationen und ebenso viele Additionen. Dabei steht das Zeichen „$\doteq$“ für „gleich bis auf Terme niedrigerer Ordnung“, d.h. wir betrachten nur den für große n entscheidenden Term mit dem größten Exponenten.

Vollkommen analog läßt sich ein gestaffeltes Gleichungssystem der Form

$$Lx = z$$

mit einer unteren Dreiecksmatrix L lösen, indem man jetzt in der ersten Zeile beginnt und sich bis zur letzten Zeile durcharbeitet. Diese Auflösung nennt man (ganz analog zur Auflösung von $Rx = z$) *Vorwärtssubstitution*.

Wir kehren nun wieder zurück zu dem allgemeinen quadratischen linearen Gleichungssystem $Ax = b$

$$\begin{array}{ccccccc} a_{11}^{(1)}x_1 & + & a_{12}^{(1)}x_2 & +\ldots+ & a_{1n}^{(1)}x_n & = & b_1^{(1)} \\ a_{21}^{(1)}x_1 & + & a_{22}^{(1)}x_2 & +\ldots+ & a_{2n}^{(1)}x_n & = & b_2^{(1)} \\ \vdots & & \vdots & & \vdots & & \vdots \\ a_{n1}^{(1)}x_1 & + & a_{n2}^{(1)}x_2 & +\ldots+ & a_{nn}^{(1)}x_n & = & b_n^{(1)} \end{array} \tag{3.5}$$

und versuchen, es in ein gestaffeltes umzuformen. Die erste Zeile muß dazu nicht verändert werden. Die restlichen Zeilen wollen wir so behandeln, daß die Koeffizienten vor x_1 verschwinden, d.h. die Variable x_1 aus den Zeilen 2 bis n *eliminiert* wird. So entsteht ein System der Art

$$\begin{array}{ccccccc} a_{11}^{(1)}x_1 & + & a_{12}^{(1)}x_2 & +\ldots+ & a_{1n}^{(1)}x_n & = & b_1^{(1)} \\ & & a_{22}^{(2)}x_2 & +\ldots+ & a_{2n}^{(2)}x_n & = & b_2^{(2)} \\ & & \vdots & & & & \vdots \\ & & a_{n2}^{(2)}x_2 & +\ldots+ & a_{nn}^{(2)}x_n & = & b_n^{(2)}\ . \end{array} \tag{3.6}$$

Haben wir das erreicht, so können wir dasselbe Verfahren auf die letzten $n-1$ Zeilen anwenden und so rekursiv ein gestaffeltes System erhalten. Es genügt daher, den ersten Eliminationsschritt von (3.5) nach (3.6) zu untersuchen. Wir setzen voraus, daß $a_{11} \neq 0$. Um den Term $a_{i1}x_1$ in Zeile $i\,(i = 2, \ldots, n)$ zu eliminieren, subtrahieren wir von der Zeile i ein Vielfaches der unveränderten Zeile 1, d.h.

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

oder explizit

$$\underbrace{(a_{i1}^{(1)} - l_{i1}a_{11}^{(1)})}_{=0} x_1 + \underbrace{(a_{i2}^{(1)} - l_{i1}a_{12}^{(1)})}_{=a_{i2}^{(2)}} x_2 + \cdots + \underbrace{(a_{in}^{(1)} - l_{i1}a_{1n}^{(1)})}_{=a_{in}^{(2)}} x_n = \underbrace{b_i^{(1)} - l_{i1}b_1^{(1)}}_{=b_i^{(2)}}.$$

Aus $a_{i1}^{(1)} - l_{i1}a_{11}^{(1)} = 0$ folgt sofort $l_{i1} = a_{i1}^{(1)}/a_{11}^{(1)}$. Damit ist der erste Eliminationsschritt unter der Annahme $a_{11}^{(1)} \neq 0$ ausführbar. Das Element $a_{11}^{(1)}$ heißt *Pivotelement*, die erste Zeile *Pivotzeile*. (Der aus dem Französischen stammende Begriff „pivot" läßt sich dabei am besten mit „Dreh- und Angelpunkt" übersetzen.) In den Zeilen 2 bis n bleibt nach diesem ersten Eliminationsschritt eine $(n-1, n-1)$-Restmatrix stehen. Wenden wir auf diese Restmatrix die Eliminationsvorschrift erneut an, so erhalten wir eine Folge

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

von Matrizen der speziellen Gestalt

$$A^{(k)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & \dots & \dots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \dots & \dots & \dots & a_{2n}^{(2)} \\ & & \ddots & & & \vdots \\ & & & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ & & & \vdots & & \vdots \\ & & & a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{bmatrix} \tag{3.7}$$

mit einer $(n-k+1, n-k+1)$-Restmatrix, auf der wir den Eliminationsschritt

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

ausführen können, wenn das *Pivotelement* $a_{kk}^{(k)}$ nicht verschwindet. Da jeder Eliminationsschritt eine lineare Operation auf den Zeilen von A ist, läßt sich der Übergang von $A^{(k)}$ und $b^{(k)}$ zu $A^{(k+1)}$ und $b^{(k+1)}$ als Multiplikation mit einer Matrix $L_k \in \mathbb{R}^{n \times n}$ von links darstellen, d.h.

$$A^{(k+1)} = L_k A^{(k)} \quad \text{und} \quad b^{(k+1)} = L_k b^{(k)} .$$

(Bei Spaltenoperationen erhielte man analog eine Multiplikation von rechts.) Die Matrix

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

ist dabei eine sogenannte *Frobenius-Matrix*. Sie hat die schöne Eigenschaft, daß die Inverse L_k^{-1} aus L_k durch einen Vorzeichenwechsel in den Elementen l_{ik} entsteht. Darüber hinaus gilt für das Produkt der L_k^{-1}, daß

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

Zusammengefaßt erhalten wir auf diese Weise das zu $Ax = b$ äquivalente gestaffelte Gleichungssystem $Rx = z$ mit

$$R = L^{-1}A \quad \text{und} \quad z = L^{-1}b.$$

Die obige Darstellung $A = LR$ der Matrix A als Produkt einer unteren Dreiecksmatrix L mit Einheitsdiagonale (d.h. $l_{ii} = 1 \ \forall \ i = 1, \ldots, n$) und einer oberen Dreiecksmatrix R heißt *$LR-$Zerlegung* von A. Wir haben gesehen

Satz 3.3 *Sei $A \in \mathbb{R}^{n\times n}$ nichtsingulär. A besitzt genau dann eine LR-Zerlegung, wenn alle führenden Hauptabschnittsdeterminanten von A ungleich Null sind.*

Dabei bezeichnet für $k \in \{1, \ldots, n\}$ $B[k]$ die k-te führende Hauptabschnittsmatrix einer Matrix $B \in \mathbb{R}^{n\times n}$, d.h.

$$B[k] = \begin{pmatrix} b_{11} & \ldots & b_{1k} \\ \vdots & & \vdots \\ b_{k1} & \ldots & b_{kk} \end{pmatrix}$$

und $\det B[k]$ die k-te führende Hauptabschnittsdeterminante von B.
Außerdem gilt, daß die LR-Zerlegung einer nichtsingulären Matrix eindeutig bestimmt ist.

Satz 3.4 *Sei $A \in \mathbb{R}^{n\times n}$ nichtsingulär. Sind $A = L_1R_1$ und $A = L_2R_2$ zwei LR-Zerlegungen von A mit unteren Dreiecksmatrizen L_1, L_2 mit Einheitsdiagonale und oberen Dreiecksmatrizen R_1, R_2, so gilt $L_1 = L_2$ und $R_1 = R_2$.*

Beweis: Aus $L_1R_1 = L_2R_2$ folgt $L_2^{-1}L_1 = R_2R_1^{-1}$. $L_2^{-1}L_1$ ist eine untere Dreiecksmatrix, die eine Einheitsdiagonale besitzt; $R_2R_1^{-1}$ ist eine obere Dreiecksmatrix, also ist $L_2^{-1}L_1 = R_2R_1^{-1} = I$, d.h. $L_2 = L_1$ und $R_2 = R_1$. $\surd$

Algorithmus *Gauß-Elimination*

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

Das *Speicherschema* für die Gauß-Elimination orientiert sich an der Darstellung (3.7) der Matrizen $A^{(k)}$. In die verbleibenden Speicherplätze können die ℓ_{ik} eingetragen werden, da Elemente mit den Werten 0 oder 1 natürlich nicht eigens gespeichert werden müssen. Der gesamte Speicheraufwand für die Gauß-Elimination beträgt $n(n+1)$ Speicherplätze, d.h. genauso viel wie für die Eingabe benötigt wird. Für den Rechenaufwand, gezählt in Multiplikationen, ergibt sich

$$\begin{aligned} \sim \sum_{k=1}^{n-1} k^2 &\doteq n^3/3 \quad \text{für a)} \\ \sim \sum_{k=1}^{n-1} k &\doteq n^2/2 \quad \text{jeweils für b) und c)} \end{aligned}$$

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

Wie man bereits an dem einfachen Beispiel

$$A = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right), \quad \det A = -1, \quad a_{11} = 0$$

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

$$A = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right) \longrightarrow \overline{A} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right) = I = LR \quad \text{mit} \quad L = R = I\,.$$

Schwierigkeiten können bei der rechnerischen Realisierung des Gaußschen Eliminationsverfahrens nicht nur bei verschwindenden, sondern bereits bei „zu kleinen“ Pivotelementen entstehen.

Beispiel 3.5 *Wir berechnen die Lösung des Gleichungssystems*

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

auf einem Rechner, der der Einfachheit halber mit nur drei Dezimalziffern arbeitet. Ergänzen wir die Zahlen durch Nullen, so erhalten wir als „exakte“ Lösung auf vier gültigen Ziffern

$$x_1 = 1.000 \qquad x_2 = 0.9999\,,$$

auf drei gültige also

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

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

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

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

Damit erhalten wir das gestaffelte System

$$\begin{array}{rcrcr} 1.00 \cdot 10^{-4} x_1 & + & 1.00 x_2 & = & 1.00 \\ & & -1.00 \cdot 10^4 x_2 & = & -1.00 \cdot 10^4 \end{array}$$

und die „Lösung“

$$x_2 = 1.00 \quad \textit{(richtig)} \qquad x_1 = 0.00 \quad \textit{(falsch!)}\,.$$

Vertauschen wir jedoch vor der Elimination die beiden Zeilen

$$\begin{array}{lrcrcl} (\tilde{a}) & 1.00x_1 & + & 1.00x_2 & = & 2.00 \\ (\tilde{b}) & 1.00 \cdot 10^{-4}x_1 & + & 1.00x_2 & = & 1.00 \ , \end{array}$$

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

$$\begin{array}{rcrcl} 1.00x_1 & + & 1.00x_2 & = & 2.00 \\ & & 1.00x_2 & = & 1.00 \end{array}$$

sowie die „richtige Lösung“

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

Man kann zeigen

Satz 3.6 *Für jede invertierbare Matrix A existiert eine Permutationsmatrix P derart, daß eine Dreieckszerlegung*

$$PA = LR$$

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

$$\mid L \mid \leq 1 \ .$$

Dieser Satz garantiert, daß wir stets ein zum Vertauschen geeignetes Element finden können, das den Platz des Diagonalelements einnehmen kann. Da es in der Regel sogar mehrere zur Vertauschung geeignete Elemente gibt, können wir unter den möglichen Vertauschungskandidaten einen solchen wählen, der insgesamt die Elimination günstig beeinflußt. Dazu betrachten wir die auszuführenden Grundoperationen, die alle die Gestalt

$$a_{jk}^{(i+k)} = a_{jk}^{(i)} - \frac{a_{j,i+1}^{(i)}}{a_{i+1,i+1}^{(i)}} \, a_{i+1,k}^{(i)}, \quad j = i+2, i+3, \ldots, n, \quad k = i+2, i+3, \ldots, n \tag{3.8}$$

besitzen. Haben wir jetzt ein Diagonalelement $a_{i+1,i+1}^{(i)}$ gewählt, das sich nur wenig von Null unterscheidet, so wird der angegebene Quotient im Verhältnis zu $a_{jk}^{(i)}$ sehr groß, und die Differenzbildung kann im Rechner nicht genau erfolgen. Es entstehen Rundungsfehler. Wählen wir dagegen das Diagonalelement möglichst groß, so bleiben die Multiplikatoren von $a_{i+1,k}^{(i)}$ unterhalb Eins, und die Größenordnung der beiden Summanden von (3.8) bleibt in etwa unverändert. Aus diesen Gründen ist es also sinnvoll, auch dann das Diagonalelement $a_{i+1,i+1}^{(i)}$ durch ein anderes zu ersetzen, wenn es nicht Null ist.

Aus den bisherigen Überlegungen ergeben sich i.w. drei Strategien zur Wahl eines neuen Diagonalelements. Das aktuell zur Elimination benutzte Diagonalelement heißt *Pivotelement* und die Suche nach einem geeigneten Pivotelement heißt entsprechend *Pivotsuche.* Die drei Strategien angewendet auf $A^{(i)}$ lauten:

Spaltenpivotsuche : Unter den Elementen $a_{j,i+1}^{(i)}, j = i+1, i+2, \ldots, n$ wähle man ein dem Betrage nach größtes als neues Diagonalelement. Sitzt dieses Element in Zeile k_0, so vertausche man Zeile k_0 mit Zeile $i+1$.

Zeilenpivotsuche : Unter den Elementen $a_{i+1,k}^{(i)}, k = i+1, i+2, \ldots n$ wähle man ein dem Betrage nach größtes als neues Diagonalelement. Sitzt dieses Element in Spalte k_0, so vertausche man Spalte k_0 mit Spalte $i+1$.

Totale Pivotsuche : Unter den Elementen $a_{jk}^{(i)}, j, k \geq i+1$ wähle man ein dem Betrage nach größtes als neues Diagonalelement. Sitzt dieses Element in Zeile j_0 und Spalte k_0, so vertausche man die Zeile j_0 mit Zeile $i+1$ und Spalte k_0 mit Spalte $i+1$.

Bei großen Matrizen ist eine vereinfachende Variante zu den drei Strategien sinnvoll. Man suche wie angegeben, aber nur so lange, bis man ein Element gefunden hat, das einen vorgegebenen Schwellenwert $s > 0$ erreicht oder überschreitet. Diese Strategie heißt *Schwellenwertmethode*.

Es ist zu beachten, daß die angegebenen Vertauschungen von Zeilen und Spalten, die ja sehr aufwendig sein können, tatsächlich nicht ausgeführt werden müssen. Es ist dazu zweckmäßig, sich neben und unter der Matrix A je einen zusätzlichen Merkvektor zu definieren, in den man die entsprechenden Vertauschungen einträgt und den man vor der Rechnung mit $1, 2, \ldots, n$ besetzt.

Unter Verwendung eines derartigen Merkvektors sind die beiden zuerst genannten Pivotstrategien gleich aufwendig und daher gleich attraktiv. Die dritte Strategie ist aufwendiger. Bei den ersten beiden Strategien benötigen wir $n + (n-1) + (n-2) + \cdots + 1 = n(n+1)/2$ Suchschritte, im dritten Verfahren brauchen wir $n^2 + (n-1)^2 + \cdots + 1 = \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}$ Schritte.

Wir betrachten jetzt einige Beispiele, die deutlich machen, daß auch in einfachen Fällen immer eine Pivotsuche eingeschaltet werden sollte. Wir benutzen durchgehend sechsstellige Arithmetik.

Beispiel 3.7 *Lösung eines linearen Gleichungssystems*
Wir behandeln $A^{(0)}x = b^{(0)}$ zuerst ohne Pivotsuche mit

$$A^{(0)} = \begin{pmatrix} 11 & 44 & 1 \\ 0.1 & 0.4 & 3 \\ 0 & 1 & -1 \end{pmatrix}, b^{(0)} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Die exakte Lösung ist

$$x^T = \frac{1}{329}(-1732, 438, 109) = (-5.26444, 1.33131, 0.331307).$$

1. Eliminationsschritt:

Wir subtrahieren (0.1/11)*-mal die erste Zeile von der zweiten Zeile und erhalten:*

$$A^{(1)} = \begin{pmatrix} 11 & 44 & 1 \\ 0 & 2.98023 * 10^{-8} & 2.99091 \\ 0 & 1 & -1 \end{pmatrix} \; ; \quad b^{(1)} = \begin{pmatrix} 1 \\ 0.990909 \\ 1 \end{pmatrix} .$$

2. Eliminationsschritt:

$$A^{(2)} = \begin{pmatrix} 11 & 44 & 1 \\ 0 & 2.98023 * 10^{-8} & 2.99091 \\ 0 & 0 & -1.00358 * 10^{8} \end{pmatrix} \; ; \quad b^{(2)} = \begin{pmatrix} 1 \\ 0.990909 \\ -3.32494 * 10^{7} \end{pmatrix} .$$

Daraus erhält man $x = (0.0607903\,,\, 0\,,\, 0.331307)^T$*, ein Vektor, der wenig mit der Lösung zu tun hat.*

Wir behandeln dasselbe Problem mit Spaltenpivotsuche:
1. Eliminationsschritt: (wie eben)

$$A^{(1)} = \begin{pmatrix} 11 & 44 & 1 \\ 0 & 2.98023 * 10^{-8} & 2.99091 \\ 0 & 1 & -1 \end{pmatrix} \; ; \quad b^{(1)} = \begin{pmatrix} 1 \\ 0.990909 \\ 1 \end{pmatrix}$$

2. Eliminationsschritt: (vertausche die Zeilen 2 und 3)

$$A^{(2)} = \begin{pmatrix} 11 & 44 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 2.99091 \end{pmatrix} \; ; \quad b^{(2)} = \begin{pmatrix} 1 \\ 1 \\ 0.990909 \end{pmatrix}$$

mit (exakter) Lösung $x^T = (-5.26444, 1.33131, 0.331307)$.

Wir haben an dem Beispiel gesehen, daß Verzicht auf Pivotsuche total falsche Ergebnisse liefern kann. Man kann auch Beispiele konstruieren, die ohne Pivotsuche auf eine singuläre Matrix stoßen und damit das Verfahren zum Abbrechen zwingen, obwohl das Gesamtsystem nicht singulär ist.

Mit der Pivotsuche sind jedoch noch nicht alle auftretenden numerischen Probleme gelöst. Zunächst wollen wir jedoch noch auf zwei Punkte hinweisen:

Bemerkung 3.8 *a) Ein singuläres Gleichungssystem wird (jedenfalls theoretisch) vom Gaußverfahren daran erkannt, daß alle Pivotsuchen erfolglos verlaufen, d.h. ein verschwindendes Pivotelement produzieren. Durch Rundungsfehler könnte es aber passieren, daß das System trotzdem als lösbar erscheint. Eine hieraus berechnete Lösung hat wenig Sinn. Man behilft sich in der Praxis üblicherweise so, daß man ein Gleichungssystem für nicht behandelbar erklärt, wenn ein Pivotelement betragsmäßig kleiner als ein vorgegebenes* ε *(z.B.* $\varepsilon = 10^{-8}$*) vorkommt.*

b) Hat man für PA eine LR-Zerlegung $PA = LR$ vorliegen, so kann man das Gleichungssystem $Ax = f$ für eine neue rechte Seite f ganz einfach lösen. Man bestimmt z mit $P^T Lz = P^T f$ durch Vorwärtseinsetzen und x mit $Rx = z$. Dann ist

$$Ax = P^T LRx = P^T Lz = P^T f.$$

Es muß also nicht wieder der gesamte Eliminationsalgorithmus durchlaufen werden.

Totale Pivotsuch ist im Vergleich zur Spaltenpivotsuche sehr zeitaufwendig und in der Praxis häufig nicht notwendig, wenn man den Zeitaufwand und den Nutzen miteinander vergleicht. Das Rundungsfehlerverhalten ist bei der Verwendung der Spaltenpivotsuche schon akzeptabel. Man zeigt in einer genauen Rückwärtsanalyse des Verfahrens folgendes:

Satz 3.9 *Sei $\hat{x}$ die mit dem Gauß-Verfahren mit Spaltenpivotsuche auf dem Rechner erzeugte Lösung von $Ax = b$. Dann gibt es eine Matrix E, so daß*

$$(A + E)\hat{x} = b$$

und für $E = (e_{ij})_{i,j \in 1,\dots,n}$ gilt:

$$|e_{ij}| \;\leq\; [c_1 n + 2c_2 n^2 + c_2^2 n^3 eps] \cdot a \cdot g \cdot eps.$$

Dabei sind $c_1,\ c_2$ Konstanten in der Größenordnung 1,

$$a = max_{i,j}|a_{ij}^{(1)}|, \quad g = \frac{1}{2} max_{i,j,k}|a_{ij}^{(k)}|/a$$

und eps ist die Maschinengenauigkeit. Für den Wachstumsfaktor g erhält man als obere Schranke $g \leq 2^{n-1}$.

Ideal wäre es, wenn der äquivalente Datenfehler E von der Größenordnung der Maschinengenauigkeit wäre. Wie weit er gegebenenfalls darüber hinaus geht, hängt von dem Wachstumsfaktor g ab. Empirische Untersuchungen zeigen g stets als moderat. Bei totaler Pivotsuche kann man zeigen, daß $g \leq 2\sqrt{k}(2 \cdot 3^{1/2} \cdot \dots \cdot k^{1/(k-1)})^{1/2}$ gilt. Diese obere Schranke ist natürlich wesentlich besser als die der Spaltenpivotsuche. Jedoch ist der Aufwand um so vieles größer und die empirischen Untersuchungen so klar für die Spaltenpivotsuche votierend, daß man dies nur in seltenen Fällen einsetzt.
Alles in allem können wir feststellen daß mit dem Gaußschen Eliminationsverfahren ein numerisch, bis auf die oben angedeutete Einschränkung, zufriedenstellendes Verfahren vorliegt.
Da das beschriebene und in der anfänglich erwähnten MATLAB-Prozedur verwendete Lösungsverfahren für lineare Gleichungssysteme also zufriedenstellend ist, haben wir noch keine Erklärung für das katastrophal falsche Ergebnis für die Hilbertmatrix gefunden. Nach unseren Betrachtungen im Kapitel 2 kann das jetzt nur noch daran liegen, daß für diese Daten das Problem der Gleichungslösung schlecht konditioniert ist, d.h. daß kleine Änderungen in den Daten, sehr große Änderungen in dem Ergebnis bewirken können. Dazu läßt sich das folgende Ergebnis beweisen.

Satz 3.10 *Sei x die Lösung von $Ax = b$ und y die Lösung von $(A + \Delta A)y = b + \Delta b$. Bezeichne δ eine Schranke für den relativen Fehler, d.h. $\frac{\|\Delta A\|}{\|A\|} \leq \delta$ und $\frac{\|\Delta b\|}{\|b\|} \leq \delta$. Dann gilt für y:*

$$\frac{\|x-y\|}{\|x\|} \leq \frac{2\delta}{1-\delta\kappa(A)}\kappa(A)$$

wobei $\kappa(A) = \|A\|\,\|A^{-1}\|$ ist.

$\kappa(A) = \|A\|\,\|A^{-1}\|$ heißt *Konditionszahl* von A. Ist diese groß , so ist die Abweichung $x - y$ groß . Für unsere Hilbertmatrix H der Größe 20 ist $\kappa(H) = 1.1896 \cdot 10^{19}$. Die vom Rechner gelieferte Lösung y ist nach unseren Überlegungen die exakte Lösung von

$$(H + E)y = b$$

und E ist in der Größenordnung der Maschinengenauigkeit, bei Matlab : $2.2204 \cdot 10^{-16}$. Nach Satz 3.4 kann dann $\|x - y\|$ von der Größenordnung $10^{-16} \cdot 10^{19} = 1000$ sein.

Die Konditionszahl $\|A\|\,\|A^{-1}\|$ einer Matrix A wird dann sehr groß, wenn A beinahe singulär ist. Denn dann existiert die Inverse A^{-1} zwar noch, hat aber typischerweise sehr, sehr große Einträge. [Betrachten Sie dazu einmal die Inverse der 20×20 Hilbertmatrix. Matlab liefert diese mittels invhilb(20).] $\|A^{-1}\|$ wird dann außerordentlich groß.

Obwohl das gewählte numerische Verfahren also in Ordnung ist, kann das Ergebnis bei sehr großer Konditionszahl von A sehr falsch sein. Gute Software wird daher neben der eigentlichen Berechnung der Lösung auch immer eine Schätzung für die Konditionszahl mitberechnen, und falls diese zu groß ausfällt, eine Warnung ausdrucken. (Bei der Inversenberechung von hilb(20) ist dieses auch geschehen.) Die Konditionszahl selbst zu berechnen ist aufwendiger als das Ausgangsproblem zu lösen, so daß man sich mit einer gut durchdachten Schätzung begnügen muß.

Zum Abschluß wollen wir uns noch überlegen, warum ein kleines Residuum $r = A\hat{x} - b$ für eine Näherungslösung nicht unbedingt besagt, daß $\hat{x}$ nahe an der Lösung x liegt. Für unsere Hilbertmatrix hatte die sehr falsche Lösung ein winzig kleines Residuum. Es gilt für das Residuum r und die Näherungslösung $\hat{x}$

$$(A - \frac{1}{\|\hat{x}\|^2} r\hat{x}^T)\hat{x} = b,$$

denn

$$\frac{1}{\|\hat{x}\|^2} r\hat{x}^T\hat{x} = \frac{1}{\|\hat{x}\|^2} r\|\hat{x}\|^2 = r.$$

$\hat{x}$ ist demnach die exakte Lösung des Problems mit der Störung $\frac{1}{\|\hat{x}\|^2} r\hat{x}^T$ in A. Es gilt

$$\|\frac{1}{\|\hat{x}\|^2} r\hat{x}^T\| = \frac{1}{\|\hat{x}\|^2}\|r\|\,\|\hat{x}\| = \frac{\|r\|}{\|\hat{x}\|}.$$

Ist also $\|r\|$ sehr klein, so auch diese Störung. Satz 3.4 sagt, daß diese Störung in die Abweichung der Lösung mit dem Faktor $\kappa(A)$ eingeht. Bei großer Konditionszahl von A kann also auch ein $\hat{x}$ mit sehr kleinem Residuum stark von x abweichen.

Bemerkung 3.11 *Hat die Systemmatrix A eine spezielle Struktur, so gilt dies auch manchmal für die Dreiecksmatrizen der LR-Zerlegung von A. Dies kann man nutzen, um die Lösungsverfahren für das System $Ax = b$ effizienter zu gestalten. Die zwei wichtigsten Beispiele sind die folgenden:*

i) Ist A symmetrisch und positiv definit, d.h. alle Eigenwerte von A sind reell und positiv, so kann man zeigen, daß A immer eine LR-Zerlegung besitzt (ohne daß man Spaltenpivotisierung durchführen muß). Man kann aber sogar erreichen, daß in

$$A = LL^T \tag{3.9}$$

zerlegt wird, wobei L eine untere Dreiecksmatrix mit reeller positiver Diagonale ist (also nicht mehr eine Einsdiagonale hat). Diese spezielle LR-Zerlegung heißt Cholesky-Zerlegung. Da man hier offensichtlich statt L und R nur eine Dreiecksmatrix ausrechnen muß, läßt sich dies günstiger durchführen als bei der allgemeinen LR-Zerlegung. Dies sieht man sogar sehr leicht, wenn man in 3.9 einfach die n^2 Gleichungen einzeln betrachtet und zwar zum Beispiel sukzessive zeilenweise. Man sieht dann, daß sich, ähnlich wie beim Rückwärtseinsetzen, die Einträge l_{ij} in L schrittweise einfach ausrechnen lassen.

ii) A heißt (m,k)-Bandmatrix, wenn alle Matrixelemente unterhalb der m-ten unteren Nebendiagonale und oberhalb der k-ten oberen Nebendiagonale verschwinden, d.h. $a_{ij} = 0$ für $i > j + k$ und $j < i + m$. Hat A eine LR-Zerlegung

$$A = LR,$$

so ist L eine $(m,0)$-Bandmatrix und R eine $(0,k)$-Bandmatrix. Diese Eigenschaft nutzt man zum Beispiel zur Entwicklung von Verfahren zur Lösung großer dünnbesetzter Matrizen, die Bandstruktur haben.

3.1.2 Iterative Lösung linearer Gleichungssysteme

Unter der iterativen Lösung linearer Gleichungssysteme $Ax = b$ mit $A \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n$ versteht man Verfahren, welche die Lösung als Grenzwert einer unendlichen Folge von Näherungslösungen berechnen; im Gegensatz dazu versteht man unter einer direkten Lösung linearer Gleichungssystme Verfahren, welche (bei rundungsfehlerfreier Rechnung) die exakte Lösung nach endlich vielen Schritten berechnen (z.B. das Gaußsche Eliminationsverfahren). Dabei verändern direkte Verfahren typischerweise die gegebene Koeffizientenmatrix und transformieren das ursprüngliche Problem in ein einfacher zu lösendes. Iterative Verfahren berechnen sukzessive Näherungslösungen an die gesuchte Lösung. Ein Iterationsschritt besteht häufig in der Ausführung einer Matrix-Vektor-Multiplikation. Während direkte Verfahren zumindest theoretisch die exakte Lösung des Problems berechnen, sind bei iterativen Verfahren Fragen der Konvergenz und Konvergenzgeschwindigkeit von Bedeutung.

Die klassischen Iterationsverfahren beruhen auf dem Umschreiben des linearen Gleichungssystems $Ax = b$ in eine Fixpunktgleichung

$$x = Tx + f$$

mit einer geeigneten Matrix $T \in \mathbb{R}^{n\times n}$ und einem Vektor $f \in \mathbb{R}^n$. Ist die Fixpunktgleichung eindeutig lösbar, so wählt man einen Startvektor $x^{(0)} \in \mathbb{R}^n$ und bildet die Folge

$$x^{(k+1)} = Tx^{(k)} + f$$

für $k = 1, 2, \dots$. Sind alle Eigenwerte von T betragsmäßig kleiner als 1, dann konvergiert diese Folge gegen die Lösung x der Gleichung $Ax = b$. Eine Fixpunktgleichung der Form $x = Tx + f$ erhält man dabei aus $Ax = b$ typischerweise aus einer Zerlegung von A in

$$A = B - C,$$

$B, C \in \mathbb{R}^{n\times n}$, B nichtsingulär, denn dann ist $(B - C)x = b$, bzw.

$$x = B^{-1}Cx + B^{-1}b.$$

Man erhält also ein System $x = Tx + f$ mit $T = B^{-1}C$ und $f = B^{-1}b$.
Die Konvergenzgeschwindigkeit einer Fixpunktiteration hängt von den Eigenwerten von T ab. Nur wenn alle Eigenwerte von T betragsmäßig kleiner als 1 sind, d.h. wenn

$$\rho(T) := \max\{|\lambda|, \lambda \text{ Eigenwert von } T\} < 1,$$

tritt Konvergenz ein. Je kleiner $\rho(T)$ ist, desto schneller ist die Konvergenz. Falls $\rho(T)$ nahe bei 1 ist, so ist die Konvergenz recht langsam und man kann versuchen, mittels Relaxation oder polynomielle Konvergenzbeschleunigung aus der gegebenen Fixpunktiteration eine schneller konvergierende herzuleiten.

Die prominentesten Vertreter dieser Iterationsverfahren sind das Jacobi-Verfahren, das Gauß-Seidel-Verfahren und das SOR-Verfahren. Diese werden im folgenden kurz beschrieben.
Das Jacobi-Verfahren, auch Gesamtschrittverfahren genannt, ist ein einfaches iteratives Verfahren zur Lösung eines linearen Gleichungssystems $Ax = b$, mit einer Matrix $A \in \mathbb{R}^{n\times n}$, deren Diagonaleinträge alle ungleich Null sind. Zerlegt man die Matrix A in die Summe des unteren Dreieckes L

$$L = \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ a_{21} & 0 & \cdots & \cdots & 0 \\ a_{31} & a_{32} & \ddots & 0 & \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{n,n-1} & 0 \end{bmatrix},$$

des oberen Dreieckes R

$$R = \begin{bmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & 0 & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & & \ddots & a_{n-1,n} \\ 0 & 0 & \cdots & \cdots & 0 \end{bmatrix},$$

und der Diagonal $D = \text{ diag }(a_{11}, a_{22}, \ldots, a_{nn})$,

$$A = L + D + R,$$

dann lautet die Fixpunktiteration des Jacobi-Verfahrens

$$Dx^{(k+1)} = -(L + R)x^{(k)} + b,$$

bzw. für $i = 1, 2, \ldots, n$

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

Das Jacobi-Verfahren konvergiert u.a. für strikt diagonaldominante Matrizen A, d.h. für Matrizen A mit

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|.$$

Man verwendet bei der Berechnung einer neuen Komponente der Näherungslösung nicht die bereits verfügbaren neuen Komponenten des Iterationsschritts. Es wird z.B. $x_1^{(k)}$ bei der Berechnung von $x_2^{(k+1)}$ verwendet, obwohl $x_1^{(k+1)}$ schon bekannt ist. Modifiziert man das Jacobi-Verfahren so, daß man stets die bereits berechneten neuen Komponenten direkt verwendet, erhält man das Gauß-Seidel-Verfahren (auch Einzelschrittverfahren genannt). Hier zerlegt man die Matrix A wieder in die Summe des unteren Dreieckes L, des oberen Dreieckes R und der Diagonal D,

$$A = L + D + R.$$

Die Fixpunktiteration des Gauß-Seidel-Verfahrens lautet

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

bzw. für $i = 1, 2, \dots, n$

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

Man verwendet hier also, im Gegensatz zum Jacobi-Verfahren, bei der Berechnung einer neuen Komponente der Näherungslösung die bereits verfügbaren neuen Komponenten des Iterationsschritts. Im allgemeinen sind diese neuen Werte genauer als die der vorgehenden Iterierten. Außerdem ist dieses Vorgehen günstig in bezug auf Speicherplatzbedarf, da man den bisherigen Wert einer Komponente der Näherungslösung direkt mit dem neu berechneten Wert überspreichern kann. Das Gauß-Seidel-Verfahren konvergiert u.a. für symmetrische, positiv definite Matrizen A.

Zur Konvergenzbeschleunigung verwendet man häufig die Technik der Relaxation. Anstelle von $x^{(k+1)} = Tx^{(k)} + f$ verwendet man als Iterationsvorschrift

$$\begin{aligned} x^{(k+1)} &= (1 - \omega)x^{(k)} + \omega(Tx^{(k)} + f) \\ &= (I + \omega(T - I))x^{(k)} + \omega f. \end{aligned}$$

Eine Fixpunktiteration $x^{(k+1)} = Tx^{(k)} + f$ konvergiert genau dann gegen die Lösung des Gleichungssystems $x = Tx + f$, wenn der Spektralradius $\rho(T)$ kleiner als 1 ist. Man versucht daher nun ω so zu bestimmen, daß schnellst mögliche Konvergenz vorliegt, d.h.

$$\min_{\omega} \{\rho(I + \omega(T - I)) < 1\}.$$

Dieser Ansatz führt beim Gauß-Seidel-Verfahren auf die Iteration

$$(D + \omega L)x^{(k+1)} = ((1 - \omega)D - \omega R)\, x^{(k)} + \omega b.$$

Man spricht dann vom Verfahren der sukzessiven Overrelaxation (SOR-Verfahren). Der maximale Bereich aus dem ω gewählt werden kann, ist das Intervall $]0, 2[$. In einigen (für die Anwendungen wichtigen) Fällen ist bekannt, für welche Wahl von ω diese Iteration optimal konvergiert. Es ist im allgemeinen vorteilhaft, Relaxationsparameter $\omega > 1$ zu betrachten.

Die bisher betrachteten Iterationsverfahren sind zum Teil schon sehr lange bekannt. Modernere Iterationsverfahren sind häufig Krylov-Raum-basierte Verfahren. Dabei wird, ausgehend von einem (beliebigen) Startvektor $x^{(0)}$, eine Folge von Näherungsvektoren $x^{(k)}$ an die gesuchte Lösung x gebildet. $x^{(k)}$ wird dazu aus einem verschobenen Krylov-Raum $\{x^{(0)}\} + \mathcal{K}_k(B, r^{(0)})$ mit

$$\mathcal{K}_k(B, r^{(0)}) = \{r^{(0)}, Br^{(0)}, B^2 r^{(0)}, \ldots, B^{k-1} r^{(0)}\}$$

und $r^{(0)} = b - Ax^{(0)}$ so gewählt, daß eine Bedingung der Art

$$b - Ax^{(k)} \perp \mathcal{L}_k$$

erfüllt ist. Die verschiedenen Versionen von Krylov-Raum-Verfahren unterscheiden sich in der Wahl der Matrix B und des k-dimensionalen Raums $\mathcal{L}_k$.
Das älteste Verfahren dieser Klasse ist das konjugierte Gradientenverfahren, welches lineare Gleichungssysteme $Ax = b$ mit symmetrisch positiv definiten Matrizen $A \in \mathbb{R}^{n \times n}$ löst. Die Idee des Verfahren beruht auf der Feststellung, daß die exakte Lösung x von $Ax = b$ gerade die Lösung der Minimierungsaufgabe

$$\min_{z \in \mathbb{R}^n} F(z)$$

mit

$$F(z) = \frac{1}{2} z^T A z - b^T z$$

ist. Zur Lösung dieser Minimierungsaufgabe berechnet man ausgehend von einem beliebigen Startvektor $x^{(0)}$ eine Folge von Näherungslösungen $x^{(1)}, x^{(2)}, \ldots$. Dazu führt man im $(k+1)$-ten Schritt eine $(k+1)$-dimensionale Minimierung

$$\min_{u_0, \ldots, u_k \in \mathbb{R}} F(x^{(k)} + u_0 r^{(0)} + \ldots + u_k r^{(k)})$$

durch. Dabei ist $r^{(j)} = b - Ax^{(j)}$, und man setzt dann $x^{(k+1)} = x^{(k)} + u_0 r^{(0)} + \ldots + u_k r^{(k)}$. Die Berechnung der $x^{(k+1)}$ ist dabei recht einfach:

Wähle $x^{(0)} \in \mathbb{R}^n$.
Setzte $r^{(0)} = p^{(0)} = b - Ax^{(0)}$.
Iteriere für $k = 0, 1, \ldots$
    Falls $p^{(k)} = 0$
    Dann stop, $x^{(k)}$ ist gesuchte Lösung.
    Sonst berechne
        $\alpha_k = \frac{(r^{(k)})^T r^{(k)}}{(p^{(k)})^T A p^{(k)}}$
        $x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$
        $r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)}$
        $\beta_k = \frac{(r^{(k+1)})^T r^{(k+1)}}{(r^{(k)})^T r^{(k)}}$

$$p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)}$$

Pro Iterationsschritt sind nur eine Matrix-Vektor-Multiplikation $Ap^{(k)}$ und einige Skalarprodukt durchzuführen; der Gesamtaufwand ist daher insbesondere für sparse Matrizen A sehr gering. Bei der Berechnung der $(k+1)$-ten Iterierten wird lediglich Information aus dem k-ten Schritt benötigt. Der Speicheraufwand ist also konstant und wächst nicht mit k an. Theoretisch ist bei exakter Rechnung spätestens $r^{(n)} = 0$ und damit $x^{(n)}$ die gesuchte Lösung. Doch aufgrund von Rundungsfehlern ist das berechnete $r^{(n)}$ in der Regel von 0 verschieden. In der Praxis setzt man das Verfahren einfach solange fort, bis $r^{(k)}$ genügend klein ist.
Der Name konjugiertes Gradientenverfahren für das beschriebene Vorgehen beruht auf den folgenden beiden Beobachtungen: Die Residuen $r^{(j)} = b - Ax^{(j)}$ lassen sich als Gradienten der Funktion F auffassen: $r^{(j)} = \text{grad } F(x^{(j)})$. Die Richtungsvektoren $p^{(k)}$ sind paarweise konjugiert, d.h. $(p^{(k)})^T A p^{(j)} = 0$ für $j = 1, 2, \ldots, k-1$.
Das konjugierte Gradientenverfahren ist ein Krylov-Raum-Verfahren zur Lösung eines symmetrisch positiv definiten Gleichungssystems. Für die Näherungslösungen $x^{(k)}$ gilt

$$x^{(k)} \in \{x^{(0)}\} + \mathcal{K}_k(A, r^{(0)}),$$

wobei der Krylov-Raum $\mathcal{K}_k(A, r^{(0)})$ definiert ist als

$$\mathcal{K}_k(A, r^{(0)}) = \text{span}\{r^{(0)}, Ar^{(0)}, A^2 r^{(0)}, \ldots, A^{k-1} r^{(0)}\}.$$

Die Residuen $r^{(k)}$ und die Richtungsvektoren $p^{(k)}$ spannen jeweils gerade diesen Krylov-Raum auf

$$\begin{aligned} \mathcal{K}_k(A, r^{(0)}) &= \text{span}\{r^{(0)}, r^{(1)}, \ldots, r^{(k-1)}\} \\ &= \text{span}\{p^{(0)}, p^{(1)}, \ldots, p^{(k-1)}\}. \end{aligned}$$

Weiter gilt

$$r^{(k)} \perp \mathcal{K}_k(A, r^{(0)}),$$

d.h. die $r^{(k)}$ bilden eine orthogonale Basis des Krylov-Raums $\mathcal{K}_k(A, r^{(0)})$.
Das konjugierte Gradientenverfahren läßt sich auch als ein Verfahren interpretieren, welches spaltenweise eine Matrix $Q_k \in \mathbb{R}^{n \times k}$ mit orthonormalen Spalten berechnet, welche A auf eine $k \times k$ Tridiagonalmatrix projiziert; d.h.

$$AQ_k = Q_k \begin{bmatrix} \gamma_1 & \delta_1 & & & \\ \delta_1 & \gamma_2 & \delta_2 & & \\ & \delta_2 & \ddots & \ddots & \\ & & \ddots & \ddots & \delta_{k-1} \\ & & & \delta_{k-1} & \gamma_k \end{bmatrix} + r_{k+1} e_k^T.$$

Für das Konvergenzverhalten des Verfahrens, d.h. für eine Aussage, wie schnell die Iterierten $x^{(k)}$ gegen die gesuchte Lösung x konvergieren, ist die Konditionszahl von A von entscheidender Bedeutung. Hat A eine kleine Konditionszahl, so wird die Konvergenz i.a. recht schnell sein. Ist die Konditionszahl von A hingegen groß, so wird die Konvergenz nur sehr langsam sein. In diesem Fall sollte man versuchen, die Konvergenzeigenschaften durch Vorkonditionierung zu verbessern. Man versucht dabei, eine symmetrisch positiv definite Matrix $C \in \mathbb{R}^{n\times n}$ zu finden so, daß das zu $Ax = b$ äquivalente Gleichungssystem $\widetilde{A}\widetilde{x} = \widetilde{b}$ mit $\widetilde{A} = C^{-1}AC^{-1}$, $\widetilde{x} = Cx$, und $\widetilde{b} = C^{-1}b$ schnellere Konvergenz liefert. Im Prinzip kann das konjugierte Gradientenverfahren nun direkt auf das vorkonditionierte Gleichungssystem $\widetilde{A}\widetilde{x} = \widetilde{b}$ angewendet werden. Dann ist am Schluß aus der resultierenden Näherungslösung $\widetilde{x}$ die Näherungslösung x des gegeben Systems $Ax = b$ durch Lösen eines Gleichungssystems $Bx = \widetilde{x}$ zu bestimmen. Es ist jedoch üblich und zweckmäßiger, das konjugierte Gradientenverfahren so umformulieren, daß direkt mit den gegebenen Daten A, b und C gerechnet wird und eine Folge von Näherungslösungen $x^{(k)}$ erzeugt wird, welche Näherungen an die gesuchte Lösung x darstellen. Man erhält dann das vorkonditionierte konjugierte Gradientenverfahren:

Wähle $x^{(0)} \in \mathbb{R}^n$.
Bestimme C. Setzte $M = C^2$.
Setze $r^{(0)} = b - Ax^{(0)}$.
Löse $Mz^{(0)} = r^{(0)}$.
Setze $p^{(1)} = z^{(0)}$.
Iteriere für $k = 1, \ldots$
  Falls $r^{(k)} = 0$
  Dann stop, $x^{(k)}$ ist gesuchte Lösung.
  Sonst berechne
$$\beta_k = \frac{(r^{(k-1)})^T z^{(k-1)}}{(r^{(k-2)})^T z^{(k-2)}}$$
$$p^{(k)} = z^{(k-1)} + \beta_k p^{(k-1)}$$
$$\alpha_k = \frac{(r^{(k-1)})^T z^{(k-1)}}{(p^{(k)})^T A p^{(k)}}$$
$$x^{(k)} = x^{(k-1)} + \alpha_k p^{(k)}$$
$$r^{(k)} = r^{(k-1)} - \alpha_k A p^{(k)}$$
$$Mz^{(k)} = r^{(k)}$$

Ein wesentlicher Punkt bei der Herleitung des konjugierten Gradientenverfahrens ist die Restriktion auf symmetrisch positiv definite Matrizen. Um das konjugierte Gradientenverfahren für beliebige (unsymmetrische oder symmetrische, nicht positiv definite) Matrizen zu verallgemeinern, wurden zahlreiche Ansätze vorgeschlagen, auf die hier nicht eingegangen werden kann. Stichworte sind das CGNR-Verfahren (Conjugate Gradient applied to Normal equations minimizing the Residual), das GCR-Verfahren (Generalized Conjugate Residual), das GMRES-Verfahren (Generalized Minimal RESidual), Das BiCG-Verfahren (BiConjugate Gradient-Verfahren) und das QMR-Verfahren (Quasi Minimal Residual-Verfahren). In der Literatur existieren zahlreiche weitere Varianten von Krylov-Raum-basierten Verfahren. Jedes dieser Verfahren hat gewisse Vor- und Nachteile, für jedes Verfahren lassen sich Beispiele finden, für welche das jeweilige Verfahren besonders gut oder besonders schlecht geeignet ist. Eine allgemeine Regel, welches Verfahren wann angewendet werden sollte, gibt es (noch) nicht.

3.2 Lineares Ausgleichsproblem und Rangbestimmung

In (fast) allen Wissenschaftsbereichen begegnet man dem Problem, unbekannte Parameter eines funktionalen Zusammenhanges bestimmen zu müssen, dessen Struktur (abgesehen von den Werten der Parameter) etwa aus Naturgesetzen oder Modellannahmen bekannt ist.
Seit Ihrer ersten bewußt erlebten Kinderkrankheit wissen Sie zum Beispiel (wenn Sie damals beim Fiebermessen gut acht gegeben haben), daß die Höhe einer Flüssigkeitssäule h, eine lineare (genauer: affin lineare) Funktion der Temperatur T ist:

$$h(T) = \alpha + \beta T. \tag{3.10}$$

Soll dieser Zusammenhang zur Vorhersage einer Höhe $\hat{h}$ bei einer vorgegebenen Temperatur $\hat{T}$ verwenden werden, müßen die reellen Parameter α und β bekannt sein. Hat dies für die von Ihnen betrachtete Flüssigkeit noch niemand vorher gemacht, so müssen α und β aus Messungen bestimmen werden.
Offenbar genügt zur Bestimmung dieser Parameter die (exakte) Kenntnis zweier Höhen h_1 und h_2 zu verschiedenen Temperaturen T_1 und $T_2, T_1 \neq T_2$. Mit (3.10) ergibt sich dann ein lineares Gleichungssystem

$$\begin{pmatrix} 1 & T_1 \\ 1 & T_2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} \tag{3.11}$$

mit regulärer Systemmatrix ($\det \left(\begin{smallmatrix} 1 & T_1 \\ 1 & T_2 \end{smallmatrix}\right) = T_2 - T_1 \neq 0$), welches α und β eindeutig bestimmt.
Leider werden die Werte h_1, h_2, T_1, T_2 in der Praxis nie exakt vorliegen, sondern mittels Messungen gewonnen mit unvermeidlichen Meßfehlern behaftet sein. Dies zieht falsche Werte für α und β und damit eine falsche Voraussage $\hat{h}$ für $h(\hat{T})$ nach sich.
In dieser Situation werden häufig weitere Messungen vorgenommen, in der Hoffnung, aus einer größeren Zahl von Meßwerten

$$(T_i, h_i), \qquad i = 1, \ldots, m, \quad m \geq 3,$$

mehr Information über die wahren Werte von α und β erschließen zu können.
Man sieht sich dann mit der Aufgabe konfrontiert, eine Gerade $\alpha + \beta T$ möglichst gut durch die erhaltenen Meßpunkte zu legen. Dazu wird die Gerade $\alpha + \beta T$ so gewählt, daß der Abstand der durch die Beobachtungen festgelegten Punkte im $\mathbb{R}^2$ von der durch die Gerade definierten Kurve im $\mathbb{R}^2$ für ein vorgegebenes Maß minimiert wird. Siehe Abbildung 3.1.
Das (3.11) entsprechende Gleichungssystem

$$\begin{pmatrix} 1 & T_1 \\ 1 & T_2 \\ \vdots & \vdots \\ 1 & T_m \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_m \end{pmatrix} \tag{3.12}$$

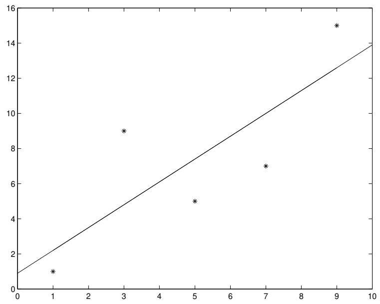


Abbildung 3.1: Ausgleichsgerade

ist in diesem Fall ($m \geq 3$) überbestimmt und hat, wenn die Messungen nicht exakt sind (oder die unexakten Messungen nicht zufällig auf einer Geraden liegen), i.a. keine Lösung. Man betrachtet nun den *Defektvektor*

$$d(\alpha,\beta) := \begin{pmatrix} 1 & T_1 \\ \vdots & \vdots \\ 1 & T_m \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} - \begin{pmatrix} h_1 \\ \vdots \\ h_m \end{pmatrix}$$

und versucht, α und β so zu wählen, daß $d(\alpha,\beta)$ möglichst klein wird. Da die Größe eines Vektors typischerweise mittels einer Norm gemessen wird, wird versucht, $\alpha^\star$ und $\beta^\star$ so zu bestimmen, daß eine Norm des Defektes, $||d(\alpha,\beta)||$, minimal wird. Hat (3.12) eine exakte Lösung, so ergibt sich $||d(\alpha,\beta)|| = 0$. Wählt man als Norm speziell die euklidische Norm, so hat man

$$||d(\alpha,\beta)||_2^2 = || \begin{pmatrix} 1 & T_1 \\ \vdots & \vdots \\ 1 & T_m \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} - \begin{pmatrix} h_1 \\ \vdots \\ h_m \end{pmatrix} ||_2^2 = \sum_{i=1}^{m} (\alpha + \beta T_i - h_i)^2$$

zu minimieren. Die Terme

$$\epsilon_i := (\alpha + \beta T_i) - h_i$$

lassen sich hier als Abstand der Geraden $\alpha + \beta T$ zu den Meßpunkten (T_i, h_i) interpretieren. Man versucht also α und β so zu wählen, daß die Summe der Quadrate der Fehler ϵ_i minimal wird.

Diese Behandlung des Problems heißt die *Methode der kleinsten Quadrate.* Sie wurde erstmals von C.F. Gauß angewendet, um Bahnelemente von Planeten zu berechnen.

Allgemein geht man von folgender Problemstellung aus: Gegeben seien m Meßpunkte

$$(t_i, b_i) \quad t_i, b_i \in \mathbb{R} \quad i = 1, \ldots, m\,,$$

die z.B. die Zustände b_i eines Objektes zu den Zeiten t_i beschreiben. Wir nehmen an, daß diesen Messungen eine Gesetzmäßigkeit zugrunde liegt, so daß sich die Abhängigkeit von b von t durch eine Modellfunktion φ mit

$$b(t) = \varphi(t; x_1, \ldots, x_n)$$

ausdrücken läßt, wobei in die Modellfunktion n unbekannte Parameter x_i eingehen. Gäbe es keine Meßfehler und würde das Modell exakt die Situation beschreiben, so sollten die Parameter $x_1, \ldots, x_n$ so zu bestimmen sein, daß

$$b_i = b(t_i) = \varphi(t_i; x_1, \ldots, x_n) \quad \text{für} \quad i = 1, \ldots, m$$

gilt. Tatsächlich sind die Messungen jedoch fehlerbehaftet und auch die Modellfunktionen stellen stets nur eine ungefähre Beschreibung der Wirklichkeit dar. Daher gilt i.a. nur

$$b_i \approx \varphi(t_i; x_1, \ldots, x_n) \quad \text{für} \quad i = 1, \ldots, m\,.$$

Nun gibt es mehrere Möglichkeiten, die einzelnen Abweichungen

$$\Delta_i := b_i - \varphi(t_i; x_1, \ldots, x_n) \quad \text{für} \quad i = 1, \ldots, m$$

zu gewichten. Wahrscheinlichkeitstheoretische Überlegungen führen zu der Aufgabe, die Parameter $x_1, \ldots, x_n$ so zu bestimmen, daß $\sum_{i=1}^{m} \Delta_i^2$ minimal wird. Wir schreiben dafür auch einfach

$$\Delta^2 = \sum_{i=1}^{m} \Delta_i^2 = \min\,. \tag{3.13}$$

In der Form (3.13) sind die Fehler der einzelnen Messungen alle gleich gewichtet. Normalerweise sind die Messungen (t_i, b_i), jedoch unterschiedlich genau, sei es, weil die Meßapparatur in verschiedenen Bereichen unterschiedlich genau arbeitet, sei es, weil die Messungen einmal mit mehr und einmal mit weniger Sorgfalt durchgeführt wurden. Zu jeder einzelnen Messung b_i gehört also in natürlicher Weise eine absolute Meßgenauigkeit bzw. Toleranz δb_i. Diese Toleranzen δb_i lassen sich in die Problemformulierung (3.13) einschließen, indem man die einzelnen Fehler mit der Toleranz gewichtet, d.h.

$$\sum_{i=1}^{m} \left(\frac{\Delta_i}{\delta b_i} \right)^2 = \min\,.$$

Diese Form der Minimierung läßt auch eine vernünftige statistische Interpretation zu (gleiche Standardabweichung).

Im folgenden betrachten wir nur den Spezialfall, daß die Modellfunktion φ *linear* in x ist, d.h.

$$\varphi(t; x_i, \ldots, x_n) = a_1(t)x_1 + \cdots + a_n(t)x_n\,,$$

wobei $a_1, \ldots, a_n : \mathbb{R} \longrightarrow \mathbb{R}$ beliebige Funktionen sind. In diesem Kapitel sei $\|\cdot\|$ stets die Euklidische Norm $\|\cdot\| = \|\cdot\|_2$. Im linearen Fall läßt sich das Ausgleichsproblem daher in der Kurzform

$$\|b - Ax\| = \min$$

schreiben, wobei $b = (b_1, \dots, b_m)^T, x = (x_1, \dots, x_n)^T$ und

$$A = (a_{ij}) \in \mathbb{R}^{m\times n} \quad \text{mit} \quad a_{ij} := a_j(t_i)\,.$$

Als weitere Einschränkung betrachten wir hier nur den überbestimmten Fall $m \geq n$, d.h. es liegen mehr Daten vor als Parameter zu bestimmen sind, was im statistischen Sinn vernünftig klingt. Wir erhalten somit die Aufgabenstellung des *linearen Ausgleichsproblems*:

Suche zu gegebenem $b \in \mathbb{R}^m$ und $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$ ein $x \in \mathbb{R}^n$, so daß

$$\|b - Ax\| = \min\,. \tag{3.14}$$

Bemerkung: Ersetzt man die 2-Norm durch die 1-Norm $\|z\|_1 := \sum_{i=1}^n |z_i|$, so ergibt sich das Standardproblem der *linearen Optimierung*, das wir hier jedoch nicht behandeln, da es in den Natur- und Ingenieurwissenschaften eher selten auftritt. Es spielt aber in den Wirtschaftswissenschaften eine wichtige Rolle. Mit der Maximumsnorm anstelle der 2-Norm $\|z\|_\infty = \max_{i\in 1,\dots,n} |z_i|$, stoßen wir auf das Problem der sogenannten *Tschebyscheffschen Ausgleichsrechnung*. Dieses Problem kommt in den Naturwissenschaften des öfteren vor, wenn auch nicht so häufig wie das der Gaußschen Ausgleichsrechnung. Da die Euklidische Norm durch ein Skalarprodukt induziert ist, läßt letztere der geometrischen Anschauung mehr Spielraum.

Beispiel 3.12 *Das einleitende Beispiel der Bestimmung einer Ausgleichsgerade $h(T) = \alpha + \beta T$ zu den Daten $(T_i, h_i), i = 1, \dots, m$ fügt sich mit*

$$A := \begin{pmatrix} 1 & T_1 \\ \vdots & \vdots \\ 1 & T_m \end{pmatrix}, x := \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, b := \begin{pmatrix} h_1 \\ \vdots \\ h_m \end{pmatrix}$$

in die eben definierte Klasse der linearen Ausgleichsprobleme ein.

Beispiel 3.13 *Wird statt der Ausgleichsgerade eine Ausgleichsparabel $h(T) = \alpha + \beta T + \gamma T^2$ zu den Daten $(T_i, h_i), i = 1, \dots, m$, gewünscht, so ergibt sich das lineare Ausgleichproblem $||Ax - b||_2 = \min$ mit*

$$A := \begin{pmatrix} 1 & T_1 & T_1^2 \\ \vdots & \vdots & \vdots \\ 1 & T_m & T_m^2 \end{pmatrix}, x := \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}, b := \begin{pmatrix} h_1 \\ \vdots \\ h_m \end{pmatrix}.$$

Zur Lösung von (3.14) ist es naheliegend,

$$\|Ax - b\|^2 = \sum_{j=1}^m (b_j - \sum_{k=1}^n a_{jk} x_k)^2$$

als Funktion der Variablen $x_1, \dots, x_n$ zu betrachten und zu sehen, für welches x dieser Ausdruck minmal wird. D.h. für welches x verschwindet der Gradient

$$\frac{\partial}{\partial x_p} \sum_{j=1}^{m} \left(b_j - \sum_{k=1}^{n} a_{jk} x_k \right)^2 = 0 \quad \text{für} \quad 1 \leq p \leq n\,.$$

Dies ist äquivalent zu

$$\sum_{j=1}^{m} a_{jp} \cdot \left(b_j - \sum_{k=1}^{n} a_{jk} x_k \right) = 0 \quad \text{für} \quad 1 \leq p \leq n\,.$$

oder

$$A^T(b - Ax) = 0$$

oder

$$A^T A x = A^T b. \tag{3.15}$$

Diese Gleichungen heißen *Normalengleichungen*. Es gilt

Satz 3.14 *Der Vektor $x \in \mathbb{R}^n$ ist genau dann Lösung des linearen Ausgleichsproblems*

$$\|b - Ax\| = \min,$$

falls er die Normalengleichung

$$A^T A x = A^T b$$

erfüllt. Insbesondere ist das lineare Ausgleichsproblem genau dann eindeutig lösbar, wenn der Rang von A maximal ist, d.h. Rang $A = n$.

Geometrisch besagen die Normalengleichungen, daß $b - Ax$ eine Normale auf dem Bildraum von A ist. Dies gibt ihnen den Namen.
Da $A^T A$ eine symmetrische Matrix ist, kann die Normalgleichung mittels des Cholesky-Verfahrens gelöst werden. Bei der Lösung der Normalgleichung können numerische Probleme auftreten, wenn die Konditionszahl der Matrix $A^T A$ sehr groß ist. Die Lösung x hat dann relativ große Fehler. Zudem sind Rundungsfehler bereits bei der Berechnung von $A^T A$ und $A^T b$ unvermeidlich.
Bei diesem Ansatz der Lösung muß also zunächst $A^T A$ berechnet und dann (3.15) gelöst werden. Dieses Gleichungssystem ist aber häufig viel schlechter konditioniert als das Ausgangsproblem. Die Konditionszahl ist hier gerade nach den Überlegungen des letzten Kapitels $\kappa(A^T A) = \|A^T A\| \, \|(A^T A)^{-1}\|$. Für unser Ausgleichsproblem (3.14) kann man hingegen das folgende Ergebnis beweisen.

Satz 3.15 *A habe vollen Rang und x bzw. $\hat{x}$ seien die Lösungen der linearen Ausgleichsprobleme*

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

Sei

$$\varrho := \sqrt{\|(A^TA)^{-1}\|}$$

und $r = b - Ax$*; gelte* $\varrho\|\Delta A\| < 1$ *und habe* $A + \Delta A$ *ebenfalls vollen Rang. Dann ist*

$$\frac{\|\hat{x} - x\|}{\|x\|} \leq \frac{\sqrt{\kappa(A^TA)}}{1 - \varrho\|\Delta A\|} \left[2\frac{\|\Delta A\|}{\|A\|} + \frac{\varrho}{1 - \varrho\|\Delta A\|} \cdot \frac{\|\Delta A\|}{\|A\|} \frac{\|r\|}{\|x\|} + \frac{\|\Delta b\|}{\|A\| \, \|x\|} \right].$$

Die relativen Datenfehler verstärken sich also hier nur mit dem Faktor $\sqrt{\kappa(A^TA)}$. Durch den Übergang von (3.14) auf das Zwischenproblem (3.15) geht man also auf ein im allgemeinen schlechter konditioniertes Zwischenproblem über. Für eine kleine Störung verstärkt das Vorgehen über die Normalengleichung den Datenfehler mit dem Quadrat der Kondiyion des Problems.

Beispiel 3.16 *Sei*

$$A = \begin{pmatrix} 1 & 1 \\ 10^{-5} & 0 \\ 0 & 10^{-5} \end{pmatrix}.$$

Dann ist bei exakter Rechnung

$$A^TA = \begin{pmatrix} 1 + 10^{-10} & 1 \\ 1 & 1 + 10^{-10} \end{pmatrix}.$$

Stört man A^TA *mit*

$$\Delta(A^TA) = \begin{pmatrix} -10^{-10} & 0 \\ 0 & -10^{-10} \end{pmatrix},$$

so ist $A^TA + \Delta(A^TA)$ *schon singulär und anstelle von* $\|b - Ax\|_2 = \min$ *wird* $\|b - (A + \Delta A)x\|_2 = \min$ *gelöst mit*

$$\Delta A = \begin{pmatrix} 0 & 0 \\ -10^{-5} & 0 \\ 0 & -10^{-5} \end{pmatrix}, \quad \textit{d.h.} \quad A + \Delta A = \begin{pmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Die Störung von A^TA *war recht klein, aber sie impliziert die große Störung* ΔA *in* A*. Ist die Maschinengenauigkeit* 10^{-10}*, so erhöht man bei einem Verfahren, das* $\|Ax - b\| = \min$ *direkt bearbeitet, noch eine vernünftige Lösung, mit dem Umweg über (3.15) aber nicht mehr.*

Aus den genannten Gründen wird man bei der numerischen Lösung von (3.14) den Umweg über die Normalengleichungen (3.15) vermeiden. Das am häufigsten verwendete Verfahren zur Lösung von (3.14) benutzt die folgenden Ergebnisse: Jede $(m \times n)$-Matrix A läßt sich zerlegen in

$$A = QR, \tag{3.16}$$

wobei Q eine orthogonale $(m \times m)$-Matrix ist, d.h. $Q^TQ = I$, und R ist eine $(m \times n)$-obere Dreiecksmatrix. In unserem Fall hatten wir $m > n$ angenommen, so daß R von der Form

$$R = \begin{pmatrix} \star & \star & \star & \dots & \star \\ & \star & \star & \dots & \star \\ & & \star & \dots & \star \\ & & & \ddots & \vdots \\ & & & & \star \\ & & & & \\ & & & & \end{pmatrix} = \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix}$$

ist, mit $\hat{R} \in \mathbb{R}^{n \times n}$ und $0 \in \mathbb{R}^{m-n \times n}$. Die Zerlegung (3.16) heißt *QR-Zerlegung von A*. Man berechnet sie analog zur LR-Zerlegung durch schrittweises Überführen von A mit elementaren orthogonalen Matrizen in obere Dreiecksform. In MATLAB kann man sich eine solche QR-Zerlegung durch den Befehl $[Q, R] = qr(A)$ berechnen lassen. Die Berechnung der QR-Zerlegung ist numerisch (rückwärts-) stabil.
Orthogonale Transformationen verändern die Länge eines Vektors nicht, d.h. für jedes $z \in \mathbb{R}^m$ ist

$$\|Q^T z\| = \sqrt{z^T Q^T Q z} = \sqrt{z^T z} = \|z\|.$$

Angenommen, $A = QR$ sei bekannt, mit Q und R wie oben. Es gilt für jedes $x \in \mathbb{R}^n$

$$\begin{aligned} \|Ax - b\|^2 &= \|Q^T Ax - Q^T b\|^2 \\ &= \|Rx - Q^T b\|^2 \\ &= \|\begin{pmatrix} \hat{R} \\ 0 \end{pmatrix} x - \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}\|^2 \\ &= \|\hat{R}x_1 - f_1\|^2 + \|f_2\|^2 \end{aligned}$$

wobei $f = Q^T b = \binom{f_1}{f_2}$ mit $f_1 \in \mathbb{R}^n, f_2 \in \mathbb{R}^{m-n}$ und $x = \binom{x_1}{x_2}$ mit $x_1 \in \mathbb{R}^n, x_2 \in \mathbb{R}^{m-n}$. Wählt man x_{opt} so ,daß

$$\hat{R}x_{opt} = f_1$$

gilt, dann wird $\|Ax - b\|$ für x_{opt} am kleinsten, denn

$$\|Ax_{opt} - b\| = \|f_2\|$$

und mehr kann hier nicht erreicht werden. Es ist also das Gleichungssystem $\hat{R}x_{opt} = f_1$ zu lösen. Da $\hat{R}$ obere Dreiecksform hat, kann dies durch einfaches Rückwärtseinsetzen geschehen. Analog zum Lösen von Gleichungssystemen mit der LR-Zerlegung wird auch hier b während der sukzessiven Transformation von A auf Dreiecksform mittransformiert, so daß $Q^T b = f$ am Ende der Zerlegungsberechnung bereits vorhanden ist. Bevor wir uns genauer ansehen, wie eine QR-Zerlegung berechnet wird, betrachten wir ein Beispiel.

Beispiel 3.17 *Wir wollen sehen, wie gut sich die Funktion*

$$y(t) = \frac{2}{\sqrt{\pi}} \int_0^t e^{-s^2} ds$$

durch ein kubisches Polynom $a_1 + a_2 t + a_3 t^2 + a_4 t^3$ *beschreiben läßt. Dazu betrachten wir zum Intervall* $[\alpha, \beta]$ *die Punkte* $(t_i, y(t_i))$, $t_0 = \alpha < t_1 < \ldots < t_m = \beta$, *und suchen* $x = (a_1, a_2, a_3, a_4)^T$, *so daß die Quadratsumme der Abweichungen,*

$$\sum_{i=1}^{m} [y(t_i) - (a_1 + a_2 t_i + a_3 t_i^2 + a_4 t_i^4)]^2 \tag{3.17}$$

minimal wird. Faßt man die y_i *zu einem Vektor* $y = (y_1, \ldots, y_m)^T$ *zusammen, so läßt sich diese Aufgabe schreiben als*

$$\|Ax - y\| = \min$$

mit

$$A = \begin{bmatrix} 1 & t_1 & t_1^2 & t_2^3 \\ 1 & t_2 & t_2^2 & t_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & t_m & t_m^2 & t_m^3 \end{bmatrix}.$$

Nun muß

$$A = QR$$

berechnet,

$$w = Q^T y$$

gebildet und für

$$R = \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix}$$

und

$$Rx = w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

die Lösung von $\hat{R}x = w_1$ *berechnet werden.*
Wir führen dies hier durch für

$$[\alpha, \beta] = [0, 2], \qquad t_i = 0.1 * i, \quad i = 0, \ldots, 20,$$

und benutzten zur Ermittlung von $y(t_i)$ *die* MATLAB-*Funktion erf. Die folgenden* MATLAB-*Befehle errechnen die Lösung:*

```
t=(0:0.1:2)                                % Berechnet t_0, ... , t_m
y=erf(t)                                   % Berechnet y(t_0), ...,y(t_m)
A = [ones(size(t',1),1) t' t.^2' t.^3']    % Berechnet  A
[Q,R]=qr(A)                                % Berechnet die QR-Zerlegung von A
w=Q' * y'
x=R(1:4,1:4)\w(1:4)                        % Berechnet R^{-1}w_1 =x
```

Die Lösung lautet auf vier Stellen

$$x = (-0.0149,\, 1.3431,\, -0.5643,\, 0.0716)^T.$$

Wir stellen die Lösung graphisch dar:

```
global x                  % x als globale Variable
plot(t,y, 'o')            % Zeichnet die Wert y(t_0), ..., y(t_m)
hold on
fplot('cub',[0,2])        % Zeichnet in dasselbe Bild die Funktion
                          %   a_1+a_2t+a_3t^2+a_4t^3
```

Dabei ist cub.m der m-file

```
function y=cub(t)
      global x;
      y=((x(4).*t+x(3)).*t+x(2)).*t+x(1);
```

Die Abbildung 3.2 zeigt die Punkte und den Kurvenverlauf. Wir sehen, daß wir mit diesen $x(i)$ eine gute Näherung an y durch das kubische Polynom erhalten.

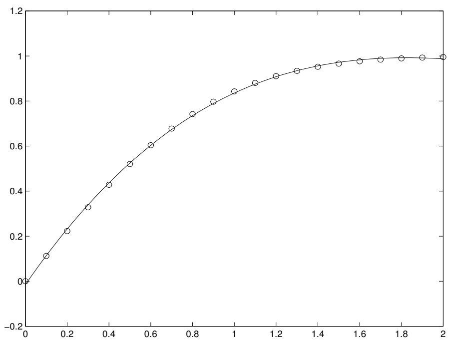


Abbildung 3.2: Approximation durch kubisches Polynom in [0, 2]

Berechnen wir eine entsprechende Lösung für

$$[\alpha, \beta] = [0, 5], \qquad t_i = 0.1 * i, \quad i = 0, \ldots, 50,$$

so erhalten wir $x = (0.0673,\, 1.0274,\, -0.3440,\, 0.0357)^T$ und sehen entsprechend in der Abbildung 3.3, daß $y(t)$ im Intervall $[0,\ 5]$ durch ein kubisches Polynom nicht gut genähert werden kann.

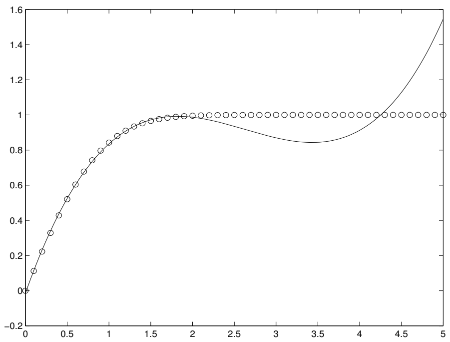


Abbildung 3.3: Approximation durch kubisches Polynom in [0, 5]

Im folgenden beschreiben wir, wie man eine QR-Zerlegung berechnet. Dies werden wir etwas allgemeiner als hier erforderlich tun, wir werden sehen, wie man die QR-Zerlegung einer Matrix $A \in \mathbb{C}^{m\times n}$ berechnet. Dabei gehen wir wie beim Gauß-Verfahren vor und eliminieren sukzessive, spaltenweise die Einträge unterhalb der Diagonalen von A. Zur Elimination verwenden wir hier elementare unitäre Matrizen, d.h. Matrizen $U \in \mathbb{C}^{m\times m}$ mit $U^H U = I$.

Man kann Matrizen der Form

$$G_{kl} := \begin{array}{cc}
\left(\begin{array}{ccccccccccc}
1 & & & & & & & & & & \\
 & \ddots & & & & & & & & & \\
 & & 1 & & & & & & & & \\
 & & & \overline{c} & & & & \overline{s} & & & \\
 & & & & 1 & & & & & & \\
 & & & & & \ddots & & & & & \\
 & & & & & & 1 & & & & \\
 & & & -s & & & & c & & & \\
 & & & & & & & & 1 & & \\
 & & & & & & & & & \ddots & \\
 & & & & & & & & & & 1
\end{array}\right)
&
\begin{array}{l}
\\ \\ \\ -k \\ \\ \\ \\ -l \\ \\ \\ \\
\end{array}
\\
\begin{array}{ccccccccccc}
 & & & | & & & & | & & & \\
 & & & k & & & & l & & &
\end{array}
&
\end{array}$$

$G_{kl} \in \mathbb{C}^{m\times m}$, mit $| c |^2 + | s |^2 = 1$ verwenden. Wie man leicht nachrechnet, ist G_{kl} unitär, d.h. $G_{kl}^H G_{kl} = I$. Im Reellen beschreibt G_{kl} eine Rotation in der (k, l)-Ebene um den Winkel θ (mit $c = \cos\theta, s = \sin\theta$). Die G_{kl} heißen *Givens-Rotationen.*

Um das Verfahren zur Reduktion von A auf obere Dreiecksgestalt mittels Givens-Rotation zu beschreiben, betrachten wir folgende **Grundaufgabe**:

> Sei $a = (a_1, \dots, a_m)^T \in \mathbb{C}^m$. Bestimme G_{kl} mit $k < l$, so daß für $(a_k, a_l) \neq (0, 0)$ in $G_{kl}a$ die l-te Komponente verschwindet.

Offenbar wird dies gelöst durch die Wahl

$$c = \frac{a_k}{\sqrt{| a_k |^2 + | a_l |^2}}, \qquad s = \frac{a_l}{\sqrt{| a_k |^2 + | a_l |^2}}, \tag{3.18}$$

denn

$$G_{kl}a = \begin{pmatrix} a_1 \\ \vdots \\ a_{k-1} \\ \overline{c}a_k + \overline{s}a_\ell \\ a_{k+1} \\ \vdots \\ a_{\ell-1} \\ ca_\ell - sa_k \\ a_{\ell+1} \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} a_1 \\ \vdots \\ a_{k-1} \\ \sqrt{\mid a_k \mid^2 + \mid a_\ell \mid^2} \\ a_{k+1} \\ \vdots \\ a_{\ell-1} \\ 0 \\ a_{\ell+1} \\ \vdots \\ a_m \end{pmatrix}.$$

Die tatsächliche Berechnung von (3.18) sollte anders erfolgen. Denn ist eine Komponente, etwa a_k, betraglich viel größer als die andere, so daß $g\ell(\mid a_\ell \mid^2 + \mid a_k \mid^2) = \mid a_k \mid^2$, so wird $g\ell(c) = 1$, und es geht wesentliche Information verloren. Besser ist, numerisch gesehen, die folgende Berechnung:

Falls $\mid a_k \mid \geq \mid a_\ell \mid$, so berechne $\tau := \frac{a_\ell}{a_k}$, $c := \frac{1}{\sqrt{1+\tau^2}}$, $s := c\tau$.
Falls $\mid a_\ell \mid > \mid a_k \mid$, so berechne $\tau := \frac{a_k}{a_\ell}$, $s := \frac{1}{\sqrt{1+\tau^2}}$, $c := s\tau$.
Dadurch vermeidet man zugleich auch Exponentenüberlauf.

Mit Hilfe der Grundaufgabe läßt sich nun $A \in \mathbb{C}^{m\times n}$ auf obere Dreiecksform transformieren. Am Beispiel einer vollbesetzten $(5,4)$-Matrix läßt sich der Algorithmus wie folgt veranschaulichen (die Indexpaare (k,ℓ) über den Pfeilen geben die Indizes der ausgeführten Givens-Rotation G_{kl} ab):

$$A = \begin{bmatrix} \star & \star & \star & \star \\ \star & \star & \star & \star \\ \star & \star & \star & \star \\ \star & \star & \star & \star \\ \star & \star & \star & \star \end{bmatrix} \xrightarrow{(5,4)} \begin{bmatrix} \star & \star & \star & \star \\ \star & \star & \star & \star \\ \star & \star & \star & \star \\ \star & \star & \star & \star \\ 0 & \star & \star & \star \end{bmatrix} \xrightarrow{(4,3)} \begin{bmatrix} \star & \star & \star & \star \\ \star & \star & \star & \star \\ \star & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & \star & \star & \star \end{bmatrix} \xrightarrow{(3,2)} \ldots \xrightarrow{(2,1)} \begin{bmatrix} \star & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & \star & \star & \star \end{bmatrix}$$

$$\xrightarrow{(5,4)} \begin{bmatrix} \star & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & 0 & \star & \star \end{bmatrix} \xrightarrow{(4,3)} \ldots \xrightarrow{(5,4)} \begin{bmatrix} \star & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & 0 & \star & \star \\ 0 & 0 & 0 & \star \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Allgemein erhält man

Für $k = 1, ..., n$
  Für $\ell = m, m-1, ..., k+1$

Löse die Grundaufgabe für $(a_{k1}, \dots, a_{km}^T)$ und $G_{\ell,\ell-1}$.
Berechne $A := G_{\ell,\ell-1}A$.

Praktisch werden nicht die $m \times m$-Matrizen $G_{\ell,\ell-1}$ aufgestellt, sondern nur die entsprechenden Umformungen der Elemente von A in der ℓ-ten und $(\ell-1)$-ten Zeile vorgenommen.

Algorithmus: $QR-$ Zerlegung mit Givens

Für $k = 1, ..., n$
  Für $l = m, m-1, ..., k+1$
    Falls $|a_{l-1,k}| \geq |a_{l,k}|$
    Dann berechne $\tau := \frac{a_{lk}}{a_{l-1,k}}, c := \frac{1}{\sqrt{1+\tau^2}}, s := c\tau$
    Sonst berechne $\tau := \frac{a_{l-1,k}}{a_{lk}}, s = \frac{1}{\sqrt{1+\tau^2}}, c := s\tau.$
    Ende Falls
    Für $j = k+1, ..., n$
      tmp $:= a_{l-1,j}$
      $a_{l-1,j} := \overline{c}\mathtt{tmp} + \overline{s}a_{l,j}$
      $a_{lj} := -s\mathtt{tmp} + ca_{lj}$
    Ende Für j
    $a_{l,k-1} := \overline{c}a_{l-1,k} + \overline{s}a_{lk}$
    $a_{kl} := 0$
  Ende für l
Ende für k

Tabelle 3.1: $QR-$ Zerlegung mit Givens

Als Aufwand für die QR-Zerlegung einer vollbesetzten Ausgangsmatrix $A \in \mathbb{C}^{m\times n}$ erhalten wir durch sorgfältiges Zählen der Operationen:

a) $\sim n^2/2$ Quadratwurzeln und $\sim 4n^3/3$ Multiplikationen, falls $m \approx n$

b) $\sim mn$ Quadratwurzeln und $\sim 2mn^2$ Multiplikationen, falls $m \gg n$

Insgesamt hat man mit $Q^H = G_{m,m-1} \dots G_{m-1,m-2}G_{m,m-1}$

$$Q^H A = \begin{pmatrix} R \\ 0 \end{pmatrix} \quad \text{und} \quad Q^H Q = I.$$

Satz 3.18 *Hat $A \in \mathbb{C}^{m\times n}, m \geq n$ Rang$(A) = n$, so existiert eine QR-Zerlegung $A = QR$ mit $r_{ii} \in \mathbb{R}$, $r_{ii} > 0$. Diese Zerlegung ist eindeutig.*

Berechnen wir die Lösung des linearen Ausgleichsproblems $\|Ax - b\|_2 = \min$, indem wir mit Hilfe der Givens-Rotation die QR-Zerlegung der Matrix $A \in \mathbb{C}^{m\times n}, m \geq n$, berechnen, so gelangen wir zu folgendem Verfahren:

1)	$Q^H A = R$	QR-Zerlegung mit Givens-Rotationen
2)	$(b_1, b_2)^H = Q^H b$	Transformation von b
3)	$Rx = b_1$	Auflösung eines gestaffelten Systems

Eine andere Möglichkeit der Berechnung einer QR-Zerlegung besteht in der Verwendung von unitären Matrizen

$$Q = I - 2vv^H$$

mit $v^H v = 1$, sogenannten *Householder-Matrizen*. Solche Matrizen beschreiben gerade die Reflexion an der auf v senkrecht stehenden Ebene. Insbesondere hängt Q nur von der Richtung von v ab. Die Householder-Reflexionen Q haben folgende Eigenschaften:

a) Q ist *hermetisch*, d.h. $Q^H = Q$.
b) Q ist *unitär*, d.h. $QQ^H = Q^H Q = I$.
c) Q ist *involutorisch*, d.h. $Q^2 = I$.

Wenden wir Q auf einen Vektor $y \in \mathbb{C}^n$ an, so gilt

$$Qy = (I - 2vv^H)y = y - 2(v^H y)v.$$

d.h. statt einer Matrix-Vektor-Multiplikation muß nur ein Skalarprodukt und eine Skalarprodukt-Vektor-Multiplikation berechnet werden. Entsprechend ergibt sich für beliebige $A \in \mathbb{C}^{m \times n}$

$$QA = (I - 2vv^H)A = A + v(-2A^H v)^H,$$

d.h. statt einer Matrix-Matrix-Multiplikation muß nur ein Matrix-Vektor-Produkt und ein Vektor-Vektor-Produkt berechnet werden.
Die Eliminationsgrundaufgabe lautet hier:

> Für vorgegebenes $y \in \mathbb{C}^m$ bestimme $\lambda \in \mathbb{C}$ und $v \in \mathbb{C}^m$ mit $v^H v = 1$ oder $v = 0$, so daß für $Q = I - 2vv^H$ gilt: $Qy = \lambda e_1$.

Falls $y_2 = y_3 = \ldots = y_m = 0$ ist, so ist $Q = I$ die naheliegendste Wahl. Andernfalls ist

$$\lambda e_1 \; = \; y - 2(v^H y)v \in span(e_1),$$

es folgt $\qquad | \lambda |^2 = ||y||_2^2 \quad \text{und} \quad v \in span(y - \lambda e_1).$
Um Auslöschung zu vermeiden, wählt man

$$\lambda := \begin{cases} -sgn(y_1)||y||_2 & \text{falls} \quad y_1 \neq 0 \\ -||y||_2 & \text{falls} \quad y_1 = 0 \end{cases} \quad \text{und} \quad v = \frac{1}{||y - \lambda e_1||_2}(y - \lambda e_1)$$

Mit Hilfe der Householder-Reflexionen können wir nun ebenfalls eine Matrix

$$A = [A_1, \ldots, A_n] \in \mathbb{C}^{m \times n}$$

in eine obere Dreiecksmatrix transformieren, indem wir sukzessive die Elemente unterhalb der Diagonalen eliminieren. Im ersten Schritt „verkürzen“ wir die erste Spalte A und erhalten

$$A \to A' := Q_1 A = \begin{bmatrix} \alpha_1 & & & \\ 0 & & & \\ \vdots & A_2' & \dots & A_n' \\ 0 & & & \end{bmatrix},$$

wobei $Q_1 = I - 2v_1 v_1^H$ mit $v_1 = \frac{1}{||A_1 - \lambda_1 e_1||_2}(A_1 - \lambda_1 e_1)$ und $\lambda_1 = -sgn(a_{11})\|A_1\|_2$, falls $a_{11} \neq 0$, sonst $\lambda_1 = -\|A_1\|_2$.

Nach dem k-ten Schritt haben wir somit die Ausgangsmatrix A bis auf eine Restmatrix $T^{k+1} \in \mathbb{C}^{(m-k)\times n-k}$ auf obere Dreiecksgestalt gebracht.

$$A^{(k)} = \begin{bmatrix} \star & \dots & \dots & \dots & \star \\ & \ddots & & & \vdots \\ & & \star & \dots & \star \\ & & 0 & & \\ & & \vdots & T^{(k+1)} & \\ & & 0 & & \end{bmatrix}.$$

Bilden wir nun die unitäre Matrix

$$Q_{k+1} = \left[\begin{array}{c|c} I_k & 0 \\ \hline 0 & \overline{Q}_{k+1} \end{array}\right],$$

wobei $\overline{Q}_{k+1} \in \mathbb{C}^{(m-k)\times(m-k)}$ wie im ersten Schritt mit $T^{(k+1)}$ anstelle von A konstruiert wird, so können wir die nächste Subspalte unterhalb der Diagonalen eliminieren. Insgesamt erhalten wir so nach $p = \min(m-1, n)$ Schritten die obere Dreiecksmatrix

$$R = Q_p \cdots Q_1 A$$

und daher wegen $Q_i^2 = I$ die Zerlegung

$$A = QR \quad \text{für} \quad Q = Q_1 \cdots Q_p \,.$$

Für den Aufwand erhalten wir bei dieser Methode

a) $\sim 2n^2 m$ Multiplikationen, falls $m \gg n$,

b) $\sim \frac{2}{3}n^3$ Multiplikationen, falls $m \approx n$.

Das bisher beschriebene Vorgehen zur Lösung eines Ausgleichsproblems setzt voraus, daß der Rang von A n ist. Ist der Rang von $A < n$, so ist (theoretisch) in der QR-Zerlegung mit $R = \left[\begin{smallmatrix} \hat{R} \\ 0 \end{smallmatrix}\right]$ die obere Dreiecksmatrix $\hat{R}$ singulär, d.h. es muß mindestens ein Diagonalelement von $\hat{R}$ Null sein. Damit läßt sich das hier vorgestellte Verfahren nicht mehr durchführen,

denn durch die Diagonalelemente von $\hat{R}$ muß ja bei der Berechnung der Lösung geteilt werden. Aufgrund von Rundungsfehlern wird man in der Praxis anstelle eines exakt singulären $\hat{R}$ auf dem Rechner nur eine beinahe singuläre Dreiecksmatrix erhalten. Dies ist ebenfalls der Fall, wenn A zwar vollen Rang n hat, aber schon eine kleine Störung zu einem Rangabfall führen kann, d.h. A ist beinahe von kleinerem Rang als n oder anders gesagt " A liegt nahe an einer Matrix mit Rang kleiner n". Wir haben schon gesehen, daß das Lösen eines fast singulären Gleichungssystems nicht sinnvoll ist, da selbst bei dem besten numerischen Verfahren Rundungsfehlereinflüsse das Ergebnis völlig verfälschen können.
Die Frage, ob eine gegebene Matrix vollen Rang hat, oder allgemeiner, welchen Rang eine gegebene Matrix hat, tritt in der Praxis auch als seperates Problem auf. So betrachtet man zum Beispiel in der Regelungstheorie Differentialgleichungssysteme der Form

$$\dot{x} = Ax + Bu,$$

wobei $x(t) \in \mathbb{R}^n$ ein Vektor ist, der die Zustände eines Systems, zum Beispiel die Spannungen in einem elektrischen Netzwerk, zum Zeitpunkt t beschreibt, und $u(t) \in \mathbb{R}^m$ ein Vektor von Steuerungsparametern zum Zeitpunkt t ist, mit denen man das Netzwerkverhalten beeinflussen kann. In solch einem System läßt sich ein Anfangszustand in jeden beliebig gegebenen Zustand $\hat{x}$ durch die Wahl einer geeigneten Steigerung u überführen genau dann, wenn der Rang der Steuerbarkeitsmatrix $(B, AB, A^2B, \dots, A^{n-1}B)$ gerade n ist. Wie kann man eine solche Bedingung durch ein numerisches Verfahren überprüfen?
Im allgemeinen ist die Frage nach dem Rang numerisch unmöglich zu beantworten. Ist etwa $A = \left(\begin{smallmatrix} 4 & -2 \\ -6 & 3 \end{smallmatrix}\right)$ und $B = \left(\begin{smallmatrix} 2 \\ -1 \end{smallmatrix}\right)$, so ist der Rang der obigen Steuerbarkeitsmatrix gerade 1, da sowohl die beiden Spaltenvektoren von A als auch der Spaltenvektor von B in dem von $\left(\begin{smallmatrix} 2 \\ -1 \end{smallmatrix}\right)$ aufgespannten eindimensionalen Vektorraum liegen und damit auch alle Vektoren der Form A^kB. In einem numerischen Verfahren wird man aufgrund von Rundungsfehlern solche (theoretischen) linearen Abhängigkeiten nicht berechnen können, da ja schon die allerkleinste Störung in einem der Vektoren, etwa zu $\hat{B} = \left(\begin{smallmatrix} 2 \\ -1-\varepsilon \end{smallmatrix}\right)$ dazu führt, daß nun schon $(\hat{B}, A\hat{B})$ den Rang 2 hat.
Zur numerischen Bestimmung des Rangs einer Matrix verwendet man die folgenden Ergebnisse. Jede $m \times n$-Matrix A besitzt eine Zerlegung

$$A = U\Sigma V^T,$$

die *Singulärwertzerlegung* (engl. singular value decomposition, kurz: SVD), wobei U eine $m \times m$ orthogonale Matrix, V eine $n \times n$ orthogonale Matrix und Σ eine $m \times n$ diagonale Matrix ist, wobei

$$\Sigma = \begin{pmatrix} \tilde{\Sigma} \\ 0 \end{pmatrix}, \qquad m \leq n \qquad \text{bzw.} \qquad \Sigma = \begin{pmatrix} \tilde{\Sigma} & 0 \end{pmatrix}, \qquad m \geq n$$

und

$$\tilde{\Sigma} = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{pmatrix}.$$

Die Diagonalelemente $\sigma_1, \dots, \sigma_n$ sind reell und nichtnegativ und der Größe nach geordnet, $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$. Sie heißen *singuläre Werte* von A oder Singulärwerte (engl.: singular values). Die SVD ist numerisch rückwärtsstabil berechenbar, d.h. $\hat{U}\hat{\Sigma}\hat{V}^T$, die vom Rechner gelieferte Zerlegung ist die theoretisch exakte Zerlegung von

$$(A + \Delta A) = \hat{U}\hat{\Sigma}\hat{V}^T,$$

wobei $\|\Delta A\| = \mathcal{O}(eps)$, $eps =$ Maschinengenauigkeit.

MATLAB liefert die Singulärwertzerlegung von A mit dem Befehl $[U, S, V] = svd(A)$.

Seien $u_1, \dots, u_m$ und $v_1, \dots, v_n$ die Spaltenvektoren von U bzw. V. Die Singulärwertzerlegung hat folgende Eigenschaften:

- $\|A\|_2 = \sigma_1$.

  Falls $m = n$ und A nicht singulär ist, so gilt $\|A^{-1}\|_2 = \frac{1}{\sigma_n}$, also gilt für die Konditionszahl von A $\kappa(A) = \frac{\sigma_1}{\sigma_n}$.

- Ist $\sigma_n \neq 0$, so $\sigma_n = \min_{B,\mathrm{Rang}(B)<n} \|A - B\|$.

  Eine solche minimale Störung, die den Rang reduziert ist $B = \sigma_n u_n v_n^T$, d.h.

  $$\mathrm{Rang}(A - \sigma_n u_n v_n^T) \leq n - 1.$$

- A läßt sich schreiben als

  $$A = U\Sigma V^T = \sum_{i=1}^{n} \sigma_i u_i v_i^T.$$

  Für $r < n$ ist $A_r = \sum_{i=1}^{r} \sigma_i u_i v_i^T$ die beste Approximation an A vom Rang $\leq r$, d.h. Rang $A_r \leq r$ und

  $$\sigma_{r+1} = \|A - A_r\| = \min_{B,\mathrm{Rang}(B)\leq r} \|A - B\|$$

- Ist $\sigma_1 \geq \dots \geq \sigma_r > \sigma_{r+1} = \sigma_{r+2} = \dots \sigma_n = 0$, so gilt: Der Nullraum von A wird von $v_{r+1}, \dots, v_n$ aufgespannt, der Bildraum von $u_1, \dots, u_r$. Ist $\sigma_1 \geq \dots \geq \sigma_r \gg \sigma_{r+1} \geq \dots \geq \sigma_n$, und sind $\sigma_{r+1}, \dots, \sigma_n$ alle von der Größenordnung *eps* der Maschinengenauigkeit, dann liegt A nahe an einer Matrix vom Rang r. Man sagt "A hat den numerischen Rang r".

Beispiel 3.19 *Die beste Approximationseigenschaft von A_r läßt sich zum Beispiel zur Datenkompression benutzen. Ein zweidimensionales schwarz-weiß Bild kann man im Rechner speichern als eine Matrix, bei der jeder Eintrag der Grauwert des Pixels in der entsprechenden Position ist.*
In MATLAB *erhält man mittels*

```
load gatlin
image(X)
colormap(map)
```

ein schwarz-weiß Bild mit sechs bekannten Numerikern, das auf der sogenannten Gatlinburg-Tagung aufgenommen wurde. Das Bild ist als 480 × 640 Matrix der Grauwerte gespeichert.
Dieses Bildes wurde in einer 480×640 *Matrix gespeichert, also mittels* 307200 *Grauwerten. Möchte man nun dieses Bild mit weniger Speicheraufwand speichern, aber so daß man bei Bedarf das Bild noch erkennen kann, so kann man die Singulärwertzerlegung benutzen und sich* $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ *ansehen, zu dessen Speicherung man nur Platz für* $k \cdot 480 + k \cdot 640 + k = 1121k$ *Daten benötigt. Dies können wir tun mit (hier* $k = 40$*, d.h.* 44840 *Daten statt* 307200*)*

```
[u,s,v]=svd(X);                     % Berechnet die SVD von X
                                    % Dies dauert eine Weile. Geduld!
                                    % Rekonstruktion
Ak = zeros(480,640);
for i=1:40;                         % Berechnet A_40 und setzt dies wieder
   Ak=Ak+s(i,i)*u(:,i)*v(:,i)';     % um in ein 480 x 640 Feld.
end
figure
image(Ak)
colormap(map)
```

Die nachfolgenden Abbildungen zeigen das Originalbild und das komprimierte Bild.

Abbildung 3.4: Gatlinburg-Foto im Original

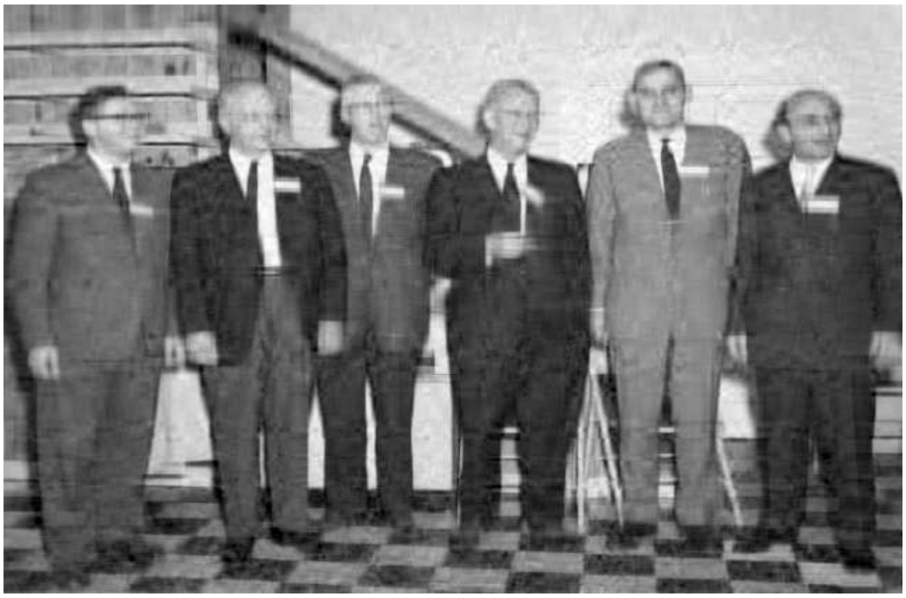


Abbildung 3.5: Gatlinburg-Foto datenreduziert

Auch das Ausgleichsproblem läßt sich mit der Singulärwertzerlegung $A = U\Sigma V^T$ lösen. Für beliebiges x gilt hier

$$\begin{aligned} \|Ax - b\| &= \|U^T AVV^T x - U^T b\| \\ &= \|\Sigma(V^T x) - (U^T b)\| \\ &= \|\Sigma y - f\| \\ &= \| \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ & 0 & \end{pmatrix} y - \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} \| \end{aligned}$$

da die Transformation mit dem orthogonalen U^T die Standard-Norm eines Vektors nicht ändert. Die kleinste Norm erhält man für y mit

$$\begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{pmatrix} y = f_1.$$

Sind die letzten Singulärwerte $\sigma_{r+1}, \ldots, \sigma_n$ zu klein, so daß A den numerischen Rang $r < n$ hat, so bestimmt man entsprechend ein y so, daß

$$\begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_r \end{pmatrix} = \begin{pmatrix} f_{11} \\ \vdots \\ f_{1r} \end{pmatrix}$$

gilt (hier sei $f_1 = (f_{11}, f_{12}, \ldots, f_{1n})^T$) und setzt die restlichen Komponenten $y_{r+1}, \ldots, y_n$ zu Null. Es gilt $x = Vy$. Dies liefert von den nun zahlreichen Lösungen des Ausgleichsproblems diejenige mit der kleinsten Norm.

Mit Hilfe der Singulärwertzerlegung läßt sich ein nützlicher Ersatz für die Inverse einer Matrix A definieren für den Fall, daß A nicht invertierbar ist. Sei $A = U\Sigma V^T$ die SVD für A. Falls $m \geq n$, so setzt man

$$\Sigma^+ = \begin{pmatrix} \frac{1}{\sigma_1} & & & \\ & \ddots & & 0 \\ & & \frac{1}{\sigma_n} & \end{pmatrix}$$

und falls $m \leq n$,

$$\Sigma^+ = \begin{pmatrix} \frac{1}{\sigma_1} & & \\ & \ddots & \\ & & \frac{1}{\sigma_n} \\ & 0 & \end{pmatrix}$$

als Pseudoinverse der Diagonalmatrix Σ (auch: Moore-Penrose-Inverse). Die Pseudoinverse von A definiert man damit als

$$A^+ = V\Sigma^+U^T.$$

Für $\sigma_n \neq 0$ und $m > n$ ist zum Beispiel damit

$$A^+A = V\Sigma^+U^TU\Sigma V^T = V\Sigma^+\Sigma V^T = VIV^T = I$$

die $n \times n$ Einheitsmatrix, und entsprechend für $m < n$ $AA^+ = I$.

$x = A^+b$ ist die Lösung von $\|Ax - b\| = \min$ mit der kleinsten Norm.

Als wichtigstes Ergebnis sollte man behalten, daß die numerisch stabile Berechnung der Singulärwerte die im allgemeinen einzige vernünftige Art ist, den (numerischen) Rang einer Matrix zu bestimmen.

3.3 Nichtlineare Gleichungssysteme

Bis jetzt haben wir uns ausschließlich mit linearen Problemen beschäftigt. In diesem Kapitel wollen wir uns der Lösung von nichtlinearen Problemen zuwenden. Beispiele dafür sind

$$\begin{aligned} \sum_{k=0}^{n} a_k x^k &= 0 \\ \sin x &= e^{-x} \end{aligned}$$

Sind f und g zwei gegebene Funktionen, so geht es also in diesem ganzen Kapitel darum, diejenigen Punkte x zu finden für die gilt:

$$f(x) = g(x).$$

Dieses Problem ist gleichbedeutend mit der Suche nach den Schnittpunkten der beiden Graphen von f und g. Diese Interpretation ist außerordentlich praktisch, weil eine Skizze der beiden Graphen häufig eine gute Information über die Anzahl und die (ungefähre) Lage der Schnittpunkte gibt. Wir nennen diese Lösungstechnik *graphische Methode.*
Nun ist es in der Regel einfacher, nur mit einer gegebenen Funktion zu arbeiten und nicht mit zweien, wie wir es oben getan haben. Ein wichtiger Sonderfall entsteht, wenn wir $g(x) = x$ setzen. In diesem Fall erhalten wir das Problem

$$f(x) = x.$$

Dieses Problem heißt *Fixpunktproblem*, weil der gesuchte Punkt x unter der Abbildung f unverändert, also *fix* bleibt. Wenn wir dieses Problem wieder graphisch interpretieren, geht es um das Auffinden der Schnittpunkte des Graphen von f mit der Winkelhalbierenden in einem rechtwinkligen Koordinatensystem. Ein zweiter wichtiger Spezialfall entsteht, wenn wir $f = 0$ setzen. In diesem Fall reduziert sich das gestellte Problem auf

$$g(x) = 0.$$

Dieses Problem heißt *Nullstellenproblem.* Hier geht es also darum, die Schnittpunkte des Graphen von g mit der x-Achse zu bestimmen. Die genannten Probleme sind im Prinzip nicht verschieden. Man kann sie auf verschiedene Weisen ineinander überführen. Durch Addition von x erhält man z.B. aus dem Nullstellenproblem $g(x) = 0$ das Fixpunktproblem $f(x) := g(x) + x = x$. Umgekehrt erhält man aus dem Fixpunktproblem $f(x) = x$ durch Subtraktion von x das Nullstellenproblem $g(x) := f(x) - x = 0$. Aus dem Fixpunktproblem $f(x) := exp(-x) = x$ kann man aber auch durch Anwendung der zu exp inversen Funktion log das Nullstellenproblem $g(x) := x + logx = 0$ gewinnen. Welche Übergänge gewählt werden, und ob das gestellte Problem in der Fixpunktformulierung oder in der Nullstellenformulierung gelöst wird, hängt von den Umständen ab. Für beide Problemformen gibt es spezielle unter etwas verschiedenen Voraussetzungen arbeitende Lösungstechniken, die wir in den nächsten Abschnitten vorstellen werden.

3.3.1 Fixpunktiteration

Wir betrachten zunächst eine nichtlineare Gleichung

$$f(x) = 0$$

mit einer beliebigen, nichtlinearen Funktion $f : \mathbb{R} \to \mathbb{R}$. Da eine Nullstelle ξ einer gegebenen Funktion f nur in wenigen Fällen in endlich vielen Schritten explizit berechnet werden kann, ist man i. a. auf Näherungsmethoden zur Lösung angewiesen. Die Idee der Fixpunktiteration besteht darin, diese Gleichung äquivalent in eine *Fixpunktgleichung*

$$\phi(x) = x$$

umzuformen und mit Hilfe der Iterationsvorschrift

$$x_{k+1} = \phi(x_k) \quad \text{mit} \quad k = 0, 1, \ldots$$

für einen gegebenen Startwert x_0 eine Folge $\{x_0, x_1, \ldots\}$ zu konstruieren. Wir hoffen, daß die so definierte Folge $\{x_k\}$ gegen einen Fixpunkt $x^\star$ mit $\phi(x^\star) = x^\star$ konvergiert, der dann auch Lösung der nichtlinearen Gleichung ist, d.h. $f(x^\star) = 0$.

Beispiel 3.20 *Wir betrachten die Gleichung*

$$f(x) := 2x - \tan x = 0. \tag{3.19}$$

Aus der Abbildung 3.6 lesen wir als Näherung für die Lösung im Intervall [0.5,2] den Wert $x^\star \approx 1.2$ ab, den wir als Startwert x_0 für eine Fixpunktiteration wählen.
Die Gleichung (3.19) können wir leicht in eine Fixpunktgleichung umformen, etwa in

$$x = \frac{1}{2} \tan x =: \phi_1(x) \quad oder \quad x = arctan(2x) =: \phi_2(x)\,.$$

Probieren wir die beiden zugehörigen Fixpunktiterationen mit dem Startwert $x_0 = 1.2$ aus, so ergeben sich folgende Zahlenwerte

k	$x_{k+1} = \frac{1}{2} \tan x_k$	$x_{k+1} = \arctan(2x_k)$
0	1.2	$\underline{1}.2$
1	1.2860	$\underline{1.1}760$
2	$1.70\ldots > \pi/2$	$\underline{1.16}87$
3		$\underline{1.16}65$
4		$\underline{1.165}8$
5		$\underline{1.165}6$
6		$\underline{1.1665}$
7		$\underline{1.1665}$

Wie man sieht, divergiert die erste Folge ($\tan x$ hat einen Pol bei $\pi/2$ und $x_2 > \pi/2$), während die zweite konvergiert. Bei der konvergenten Folge tritt ungefähr bei jedem zweiten Iterationsschritt eine gültige Dezimalziffer mehr auf.

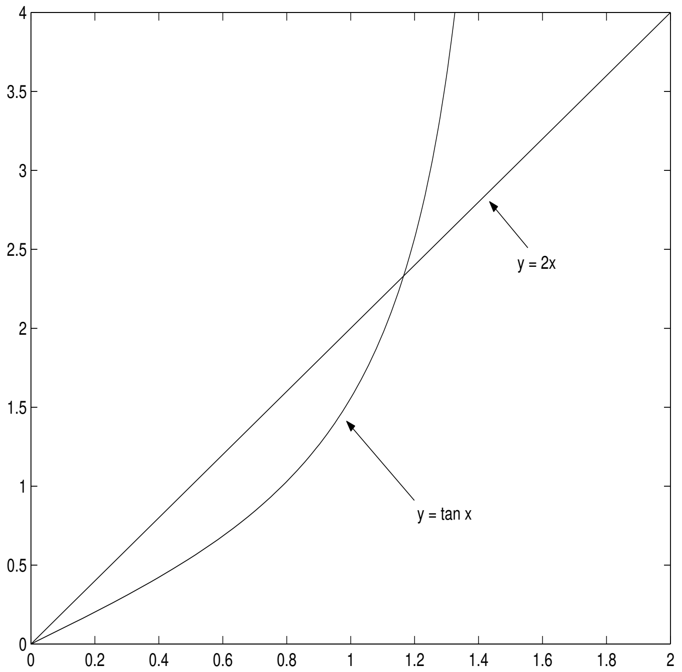


Abbildung 3.6: graphische Lösung von $2x - tanx = 0$

Offensichtlich konvergiert nicht jede naiv konstruierte Fixpunktiteration. Wir betrachten deshalb nun allgemein Folgen $\{x_k\}$, die durch eine *Iterationsfunktion* ϕ

$$x_{k+1} = \phi(x_k)$$

gegeben sind und versuchen, die Fragen

1) Wie konstruiert man geeignete Fixpunktiterationen?
2) Unter welchen Bedingungen konvergiert $\{x_j\}_j$?
3) Wie schnell konvergiert $\{x_j\}_j$?

zu beantworten.
Typischerweise betrachtet man hier die Differenz zweier Folgenglieder

$$\mid x_{k+1} - x_k \mid = \mid \phi(x_k) - \phi(x_{k-1}) \mid$$

und versucht diese Differenz durch die Differenz $\mid x_k - x_{k-1} \mid$ der vorhergehenden Folgenglieder abzuschätzen, um eine Aussage über die Konvergenz der Folge $\{x_j\}_j$ zu erhalten. Dies führt zu der folgenden theoretischen Charakterisierung:
Sei $I = [a, b] \subset \mathbb{R}$ ein Intervall und $\phi : I \to \mathbb{R}$. ϕ ist *kontrahierend auf* I, falls es ein $0 < \theta < 1$ gibt, so daß

$$\mid \phi(x) - \phi(y) \mid \leq \theta \mid x - y \mid \quad \text{für alle} \quad x, y \in I\,.$$

Für stetig differenzierbares ϕ ist die *Lipschitz-Konstante* θ leicht zu berechnen.

Lemma 3.21 *Ist $\phi : I \to \mathbb{R}$ stetig differenzierbar, $\phi \in C^1(I)$, so gilt*

$$\sup_{x,y\in I} \frac{|\ \phi(x) - \phi(y)\ |}{|\ x-y\ |} = \sup_{z\in I} \quad |\ \phi'(z)\ | < \infty\,.$$

Satz 3.22 *Sei $I = [a,b] \subset \mathbb{R}$ ein Intervall und $\phi : I \to I$ eine kontrahierende Abbildung mit Lipschitz-Konstante $\theta < 1$. Dann folgt:*

a) Es existiert genau ein Fixpunkt $x^\star$ von $\phi, \phi(x^\star) = x^\star$.

b) Für jeden Startwert $x_0 \in I$ konvergiert die Fixpunktiteration $x_{k+1} = \phi(x_k)$ gegen $x^\star$ mit

$$|\ x_{k+1} - x_k\ | \le \theta\ |\ x_k - x_{k-1}\ | \quad \textit{und} \quad |\ x^\star - x_k\ | \le \frac{\theta^k}{1-\theta}\ |\ x_1 - x_0\ |\ .$$

Satz 3.22 ist ein Spezialfall des Banachschen Fixpunktsatzes.

Zur Beurteilung der *Konvergenzgeschwindigkeit* einer Fixpunktiteration definieren wir den Begriff der *Konvergenzordnung* einer Folge $\{x_k\}$. Eine Folge $\{x_k\}, x_k \in \mathbb{R}^n$, konvergiert mit der *Ordnung (mindestens)* $p \ge 1$ gegen $x^\star$, falls es eine Konstante $C \ge 0$ gibt, so daß

$$\|x_{k+1} - x^\star\| \le C\|x_k - x^\star\|^p\,,$$

wobei wir im Fall $p = 1$ zusätzlich verlangen, daß $C < 1$. Im Fall $p = 1$ sprechen wir auch von *linearer*, für $p = 2$ von *quadratischer Konvergenz*. Weiter heißt $\{x_k\}$ *superlinear* konvergent, falls es eine nicht negative Nullfolge $C_k \ge 0$ mit $\lim_{k\to\infty} C_k = 0$ gibt, so daß

$$\|x_{k+1} - x^\star\| \le C_k\|x_k - x^\star\|\,.$$

Wie wir oben in dem einfachen Beispiel $f(x) = 2x - \tan x$ gesehen haben, muß man zur Lösung des nichtlinearen Problems $f(x) = 0$ eine geeignete Fixpunktiteration unter mehreren möglichen auswählen. Dies ist im allgemeinen keine einfache Aufgabe. Wegen

$$|\ x_{k+1} - x^\star\ | = |\ \phi(x_k) - \phi(x^\star)\ | \le \theta\ |\ x_k - x^\star\ |$$

und $0 \le \theta < 1$ konvergiert die Fixpunktiteration

$$x_{k+1} = \phi(x_k)$$

für eine kontrahierende Abbildung ϕ im allgemeinen Fall nur linear. Von einem guten Iterationsverfahren erwarten wir jedoch, daß es zumindest *superlinear* oder *linear mit einer kleinen Konstante* $C \ll 1$ konvergiert.
Im eindimensionalen Fall $f : \mathbb{R} \to \mathbb{R}$ kann man die Ordnung eines durch $\phi : \mathbb{R} \to \mathbb{R}$ erzeugten Iterationsverfahrens häufig leicht bestimmen, wenn $\phi(x)$ in einer Umgebung

$U(x^\star)$ genügend oft differenzierbar ist. Ist etwa $x_k \in U(x^\star)$ und gilt $\phi^{(j)}(x^\star) = 0$ für $j = 1, 2, \ldots, p-1$, aber $\phi^{(p)}(x^\star) \neq 0$, so folgt mittels Taylor-Entwicklung

$$x_{k+1} = \phi(x_k) = \phi(x^\star) + \frac{(x_k - x^\star)^p}{p!}\phi^{(p)}(x^\star) + O(\|x_k - x^\star\|^{p+1})$$

und

$$\lim_{k\to\infty} \frac{x_{k+1} - x^\star}{(x_k - x^\star)^p} = \frac{\phi^{(p)}(x^\star)}{p!}\,.$$

Für $p = 2, 3, \ldots$ liegt also dann ein Verfahren p-ter Ordnung vor.

3.3.2 Intervallschachtelung, Regula falsi, Sekantenmethode

Wir betrachten die Aufgabe, zu einer stetigen, nichtlinearen Funktion $f : \mathbb{R} \to \mathbb{R}$ die Lösungen der Gleichung $f(x) = 0$ zu berechnen. Das Verfahren der *Intervallschachtelung* ist durch Überlegungen der Analysis motiviert, die reelle Lösung einer Gleichung $f(x) = 0$ durch systematisch kleiner werdende Intervalle einzuschließen. Dazu geht man von der Annahme aus, es sei ein Intervall $[a, b]$ bekannt, so daß $f(a)f(b) < 0$ gilt. Aus Stetigkeitsgründen existiert dann eine Lösung s im Inneren von $[a, b]$ mit $f(s) = 0$. Für den Mittelpunkt $\mu = \frac{a+b}{2}$ des Intervalls $[a, b]$ wird der Funktionswert $f(\mu)$ berechnet. Ist $f(\mu) \neq 0$, entscheidet sein Vorzeichen, in welchem der beiden Teilintervalle $[a, \mu]$ und $[\mu, b]$ die gesuchte Lösung s ist. Ist $f(a) \cdot f(\mu) < 0$, so liegt s im Intervall $[a, \mu]$, sonst im Intervall $[\mu, b]$. Das Verfahren kann nun mit dem halbierten Intervall fortgesetzt werden, bis s im Innern eines hinreichend kleinen Intervalls liegt. Die Konvergenzordnung dieses Verfahrens ist $p = 1$.

In der Methode der *Regula falsi* geht man auch davon aus, daß ein Intervall $[x_0, x_1]$ bekannt sei mit $f(x_0)\,f(x_1) < 0$. Die beiden Startwerte x_0, x_1 können wir als Näherungen für die gesuchte Lösung s betrachten. Anstatt den Mittelpunkt des Intervalls als nächsten Näherungswert zu nehmen, bestimmt man x_2 als Nullstelle der Geraden P durch die Punkte $(x_0, f(x_0)), (x_1, f(x_1))$. Die Gerade P hat die Darstellung

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

woraus

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

folgt. Das Vorzeichen von $f(x_2)$ bestimmt, wie bei der Methode der Intervallschachtelung, das Intervall, in welchem s liegt.
Die Konvergenzordnung der Methode der Regula falsi ist für eine einfache Lösung, d.h. $f'(s) \neq 0$, und unter der Voraussetzung $f''(s) \neq 0$ gleich 1.

Als Modifikation der Regula falsi verzichtet man in der *Sekantenmethode* darauf, die Lösung s durch zwei Näherungswerte einzuschließen. Mit zwei vorzugebenden Näherungswerten x_0 und x_1, welche die Lösung s nicht einzuschließen brauchen, bestimmt man analog zur

Algorithmus: Regula falsi
`Sei` $[a,b]$ `mit` $f(a)f(b) < 0$ `gegeben.`
`Wiederhole`
$\mu = b - f(b)\frac{b-a}{f(b)-f(a)}$
`falls` $f(\mu) = 0$,
`dann stop`
`sonst falls` $f(a)f(\mu) < 0$
`dann` $b = \mu$
`sonst` $a = \mu$

Tabelle 3.2: Regula falsi

Regula falsi den Wert x_2 als Nullstelle der Geraden durch die Punkte $(x_0, f(x_0)), (x_1, f(x_1))$. Ungeachtet der Vorzeichen der Funktionswerte $f(x_0), f(x_1)$ und $f(x_2)$ wird mit der Geraden durch $(x_1, f(x_1)), (x_2, f(x_2))$ der nächste Wert x_3 bestimmt.

Algorithmus: Sekantenverfahren
`für` $k = 1, 2, \ldots$
$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$

Tabelle 3.3: Sekantenverfahren

Dies setzt voraus, daß $f(x_k) \neq f(x_{k-1})$ gilt. Das Sekantenverfahren ist ein Verfahren mit Konvergenzordnung $p = 1,618$ (falls $f(s) = 0, f'(s) = 0, f''(s) \neq 0$).

Beispiel 3.23 *Zur Berechnung der Nullstelle der Funktion*

$$f(x) = \tan(\frac{x}{4}) - 1$$

(siehe Abbildung 3.7) werden die drei Methoden angewendet. Für alle drei Verfahren wird das Startintervall $[2,\ 4]$ *verwendet, für welches* $f(2) \approx -0.4537$ *und* $f(4) \approx 0.5574$ *gilt. Der Rechengang der Methode der Intervallhalbierung ist in Tabelle 3.4 zusammengestellt (berechnet mit* MATLAB*). Die Iteration wurde abgebrochen, sobald* $|b - a| < 2.2204 * 10^{-14}$ *galt. Für* $f(a)$ *erhalten wir hier* $-5.44 * 10^{-15}$*.*
Die Methode der Regula falsi arbeitet im Vergleich zur Intervallhalbierung bedeutend effizienter. Da die Funktion $f(x)$ *im betrachteten Intervall konvex ist, bleibt die obere Intervallgrenze unverändert. Die Iteration wurde abgebrochen, sobald entweder* $f(a)$ *oder* $f(b)$ *kleiner als* $2.2204 * 10^{-14}$ *galt. Die Ergebnisse sind in Tabelle 3.5 zusammengestellt.*

Iteration	a	b	abs(b-a)
1	3.00000000000000	4.00000000000000	1.00000000000000
2	3.00000000000000	3.50000000000000	0.50000000000000
3	3.00000000000000	3.25000000000000	0.25000000000000
4	3.12500000000000	3.25000000000000	0.12500000000000
5	3.12500000000000	3.18750000000000	0.06250000000000
6	3.12500000000000	3.15625000000000	0.03125000000000
7	3.14062500000000	3.15625000000000	0.01562500000000
8	3.14062500000000	3.14843750000000	0.00781250000000
9	3.14062500000000	3.14453125000000	0.00390625000000
10	3.14062500000000	3.14257812500000	0.00195312500000
11	3.14062500000000	3.14160156250000	0.00097656250000
12	3.14111328125000	3.14160156250000	0.00048828125000
13	3.14135742187500	3.14160156250000	0.00024414062500
14	3.14147949218750	3.14160156250000	0.00012207031250
15	3.14154052734375	3.14160156250000	0.00006103515625
16	3.14157104492188	3.14160156250000	0.00003051757812
17	3.14158630371094	3.14160156250000	0.00001525878906
18	3.14158630371094	3.14159393310547	0.00000762939453
19	3.14159011840820	3.14159393310547	0.00000381469727
20	3.14159202575684	3.14159393310547	0.00000190734863
21	3.14159202575684	3.14159297943115	0.00000095367432
22	3.14159250259399	3.14159297943115	0.00000047683716
23	3.14159250259399	3.14159274101257	0.00000023841858
24	3.14159262180328	3.14159274101257	0.00000011920929
25	3.14159262180328	3.14159268140793	0.00000005960464
26	3.14159265160561	3.14159268140793	0.00000002980232
27	3.14159265160561	3.14159266650677	0.00000001490116
28	3.14159265160561	3.14159265905619	0.00000000745058
29	3.14159265160561	3.14159265533090	0.00000000372529
30	3.14159265346825	3.14159265533090	0.00000000186265
31	3.14159265346825	3.14159265439957	0.00000000093132
32	3.14159265346825	3.14159265393391	0.00000000046566
33	3.14159265346825	3.14159265370108	0.00000000023283
34	3.14159265358467	3.14159265370108	0.00000000011642
35	3.14159265358467	3.14159265364287	0.00000000005821
36	3.14159265358467	3.14159265361377	0.00000000002910
37	3.14159265358467	3.14159265359922	0.00000000001455
38	3.14159265358467	3.14159265359194	0.00000000000728
39	3.14159265358830	3.14159265359194	0.00000000000364
40	3.14159265358830	3.14159265359012	0.00000000000182
41	3.14159265358921	3.14159265359012	0.00000000000091
42	3.14159265358967	3.14159265359012	0.00000000000045
43	3.14159265358967	3.14159265358990	0.00000000000023
44	3.14159265358978	3.14159265358990	0.00000000000011
45	3.14159265358978	3.14159265358984	0.00000000000006
46	3.14159265358978	3.14159265358981	0.00000000000003
47	3.14159265358978	3.14159265358980	0.00000000000001

Tabelle 3.4: Intervallschachtelung

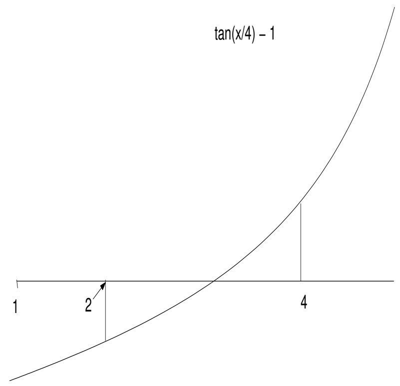


Abbildung 3.7: $\tan(\frac{x}{4})$ im Intervall $[1,\ 5]$

Iteration	a	b	f(a)	f(b)
1	2.89742886207283	4.00000000000000	-0.11519348217039	0.55740772465490
2	3.08626142352167	4.00000000000000	-0.02728986027886	0.55740772465490
3	3.12890876588468	4.00000000000000	-0.00632191841585	0.55740772465490
4	3.13867757577880	4.00000000000000	-0.00145647772687	0.55740772465490
5	3.14092230163740	4.00000000000000	-0.00033511981728	0.55740772465490
6	3.14143847851740	4.00000000000000	-0.00007708456510	0.55740772465490
7	3.14155719357547	4.00000000000000	-0.00001772984999	0.55740772465490
8	3.14158449778639	4.00000000000000	-0.00000407789339	0.55740772465490
9	3.14159077775191	4.00000000000000	-0.00000093791850	0.55740772465490
10	3.14159222214620	4.00000000000000	-0.00000021572177	0.55740772465490
11	3.14159255435756	4.00000000000000	-0.00000004961612	0.55740772465490
12	3.14159263076633	4.00000000000000	-0.00000001141173	0.55740772465490
13	3.14159264834039	4.00000000000000	-0.00000000262470	0.55740772465490
14	3.14159265238243	4.00000000000000	-0.00000000060368	0.55740772465490
15	3.14159265331210	4.00000000000000	-0.00000000013885	0.55740772465490
16	3.14159265352592	4.00000000000000	-0.00000000003194	0.55740772465490
17	3.14159265357510	4.00000000000000	-0.00000000000735	0.55740772465490
18	3.14159265358641	4.00000000000000	-0.00000000000169	0.55740772465490
19	3.14159265358902	4.00000000000000	-0.00000000000039	0.55740772465490
20	3.14159265358961	4.00000000000000	-0.00000000000009	0.55740772465490
21	3.14159265358975	4.00000000000000	-0.00000000000002	0.55740772465490
22	3.14159265358978	4.00000000000000	-0.00000000000000	0.55740772465490
23	3.14159265358979	4.00000000000000	-0.00000000000000	0.55740772465490

Tabelle 3.5: Regula falsi

Iteration	a	f(a)
-1	2.00000000000000	-0.45369751015621
0	4.00000000000000	0.55740772465490
1	2.89742886207283	-0.11519348217039
2	3.20282539232243	0.03109480332029
3	3.13791079570277	-0.00183923651108
4	3.14153602044946	-0.00002831616926
5	3.14159270570235	0.00000002605628
6	3.14159265358906	-0.00000000000037
7	3.14159265358979	-0.00000000000000

Tabelle 3.6: Sekantenverfahren

Das Sekantenverfahren liefert die Lösung in sieben Iterationsschritten, siehe Tabelle 3.6. Das Abbruchkriterium war hier dasselbe wie bei der Regula falsi.
Das Sekantenverfahren wurde nochmals mit anderen Anfangswerten gestartet, so daß beide Funktionswerte am Anfang negativ sind, die Nullstelle also nicht in dem vorgegebenen Intervall liegt (Intervall $[1, 2]$, $f(1) \approx -0.7447$, $f(2) \approx -0.4537$*), siehe Tabelle 3.7.*

Iteration	a	f(a)
-1	1.00000000000000	-0.74465807877896
0	2.00000000000000	-0.45369751015621
1	3.55930926415136	0.23416365691218
2	2.94704536091904	-0.09283181256897
3	3.12086296150474	-0.01031149980246
4	3.14258270654710	0.00049514904471
5	3.14158753115451	-0.00000256121436
6	3.14159265232182	-0.00000000063399
7	3.14159265358979	0.00000000000000

Tabelle 3.7: Sekantenverfahren mit anderen Startwerten

3.3.3 Newton-Verfahren

Ist die Funktion $f(x)$ der zu lösenden Gleichung $f(x) = 0$ stetig differenzierbar, und ist überdies die erste Ableitung ohne großen Rechenaufwand berechenbar, so wird im Verfahren von Newton die Lösung s durch den Schnittpunkt der Tangenten durch $f(x_0)$ mit der x-Achse approximiert.

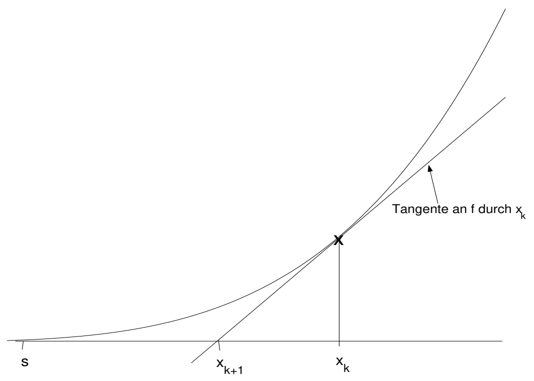


Abbildung 3.8: Newton-Verfahren

Aus der Tangentengleichung

$$y = f(x_k) + (x - x_k)f'(x_k) \quad (= 0)$$

ergibt sich so die Iterationsvorschrift

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

Die Methode von Newton gehört zur Klasse der Fixpunktiterationen mit der Funktion

$$F(x) = x - \frac{f(x)}{f'(x)} \quad \text{mit} \quad F(s) = s\,.$$

Es gilt

Satz 3.24 *Die Funktion $f(x)$ sei dreimal stetig differenzierbar in einem Intervall $I_1 = [a, b]$ mit $a < s < b$, und es sei $f(s) = 0, f'(s) \neq 0$, d.h. s sei eine einfache Nullstelle von $f(x)$. Dann existiert ein Intervall $I = [s - \delta, s + \delta]$ mit $\delta > 0$, für welches $F : I \to I$ eine kontrahierende Abbildung ist. Für jeden Startwert $x^{(0)} \in I$ ist die Folge $x^{(k)}$ des Newtonschen Verfahrens mindestens quadratisch konvergent.*

Die quadratische Konvergenz der Newtonfolge x_k macht die Methode sehr beliebt. Dennoch drängt sich ein Vergleich des Rechenaufwandes des Verfahrens von Newton mit demjenigen der Sekantenmethode auf. Dazu treffen wir die Arbeitshypothese, daß die Berechnung eines Funktionswertes $f(x)$ vergleichbar aufwendig ist wie die Berechnung der Ableitung $f'(x)$ zu gegebenem x. Ein Iterationsschritt des Verfahrens von Newton erfordert die gleichzeitige Auswertung von $f(x_k)$ und $f'(x_k)$. Mit demselben Aufwand können z w e i Schritte der Sekantenmethode ausgeführt werden.

Die Konvergenzordnung für einen Doppelschritt der Sekantenmethode ist deshalb 2.618 und somit größer als die Konvergenzordnung des Verfahrens von Newton. Unter dem Gesichtspunkt des asymptotisch gültigen Fehlergesetzes ist die Sekantenmethode effizienter.

Beispiel 3.25 *Zur Berechnung der Nullstelle der Funktion*

$$f(x) = \tan(\frac{x}{4}) - 1$$

wird hier nun das Newton-Verfahren verwendet. Der Rechengang ist in Tabelle 3.8 zusammengestellt. Die Iteration wurde abgebrochen, sobald $|f(a)| < 2.2204 * 10^{-14}$ galt. Die Konvergenz ist hier offensichtlich quadratisch. Es waren sechs Iterationsschritte nötig, d.h. es wurden je 6 Auswertungen von $f(x)$ und $f'(x)$ benötigt. Da die Berechnungen von $f(x)$ und $f'(x)$ gleich aufwendig sind, bestimmt die Sekantenmethode die Lösung s mit weniger Aufwand (mit nur 9 *Funktionsauswertungen, siehe Tabelle 3.6).*

Während die Intervallschachtelung immer gegen die gesuchte Lösung konvergiert, kann beim Newton-Verfahren auch Divergenz eintreten. Es gibt hier keine Garantie, daß x_{k+1} näher an der gesuchten Lösung liegt als x_k. Darüberhinaus bedeutet ein kleiner Wert $f'(x_k)$ eine Verlangsamung der Konvergenz, denn dann ist die Tangente an f durch $f(x_k)$ sehr steil (fast horizontal).

Iteration	a	f(a)
0	1.00000000000000	-0.74465807877896
1	3.79631404657234	0.39566265864546
2	3.25943543617547	0.06072805225952
3	3.14513155420752	0.00177101763479
4	3.14159578639006	0.00000156640136
5	3.14159265359225	0.00000000000123
6	3.14159265358979	-0.00000000000000

Tabelle 3.8: Newton-Verfahren

Beispiel 3.26 *Betrachte die Funktion*

$$f(x) = x^2 - a^2$$

für verschiedene $a \in \mathbb{R}$. *Wir haben* $f'(x) = 2x$ *und* $f(a) = 0$. *In der Nähe der Nullstelle haben wir daher, daß* f' *ungefähr* $2a$ *ist. In der Tabelle 3.9 ist die Anzahl der Iterationen des Newton-Verfahrens protokolliert, welche für verschiedene Werte von a zur Konvergenz benötigt wurden. Man sieht sehr deutlich, wie sich die Konvergenzrate von quadratisch zu*

a	Iterationen
10^{0}	5
10^{-4}	18
10^{-8}	31
10^{-12}	43
0	50

Tabelle 3.9: langsame Newton-Konvergenz

linear verschiebt je kleiner a wird (d.h. je steiler die Tangente in der Nähe der gesuchten Lösung ist).

Das folgende Beispiel zeigt eine Situation auf, in der das Newton-Verfahren divergiert.

Beispiel 3.27 *Betrachte die Funktion*

$$f(x) = \arctan(x).$$

Es gilt $f'(x) = \frac{1}{1+x^2}$ *und daher für die Newton-Iteration*

$$x_{k+1} = x_k - (1 + x_k^2)\arctan(x_k).$$

Man kann zeigen, daß für jedes $|x_k| > 1.3917$ $|x_{k+1}| > |x_k|$ *gilt. Das impliziert, daß die Iteration für jeden Startwert außerhalb des Intervalls* $[-1.3917,\ 1.3917]$ *divergiert.*

Wie schon erwähnt ist es in der Praxis meist nicht möglich, die Voraussetzungen von Satz 3.24 zu überprüfen. Daher ist eine sorgfältige Implementierung des Newton-Verfahrens notwendig, um den Abbruch des Verfahrens zu bestimmen. Eine Möglichkeit wäre die Iteration zu beenden, wenn $|x_{k+1} - x_k|$ klein genug ist. Denn es gilt falls $\lim x_k = x$, dann $\lim |x_{k+1} - x_k| = 0$. Aber die Umkehrung dieser Aussage ist nicht notwendigerweise richtig. D.h. aus $\lim |x_{k+1} - x_k| = 0$ kann man nicht unbedingt $\lim x_k = x$ schließen, wie man an folgendem Beispiel sieht: Für $f(x) = \tan x$ und $x_k = \pi/2 - 10^{-5}$, folgt $|x_{k+1} - x_k| \approx 10^{-15}$, obwohl die nächstliegende Wurzel 0 ist. Alternativ könnte man die Iteration beenden, wenn $|f(x_k)|$ klein genug ist. Dies bedeutet aber nicht, daß x_k nahe an der gesuchten Lösung liegt. Für $f(x) = (x-1)^{10}$ ist $f(1.1) = 0.1^{10}$ sehr klein, aber 1.1. ist relativ weit weg von der Lösung 1.

In jedem Iterationsschritt muß $f(x)$ und $f'(x)$ ausgewertet werden. Der Wert der ersten Ableitung ändert sich im Verfahren von Newton oft nicht mehr stark in den letzten Iterationsschritten. Um dieser Feststellung Rechnung zu tragen und um gleichzeitig die Zahl der oft aufwendigen Berechnungen der ersten Ableitung zu reduzieren, rechnet man mit dem *vereinfachten Verfahren von Newton*

$$x_{(k+1)} = x_{(k)} - \frac{f(x_k)}{f'(x_0)}, \quad (k = 0, 1, 2, \dots)\,.$$

Die erste Ableitung wird einmal für den (guten!) Startwert x_0 berechnet und für die weiteren Iterationen beibehalten. Die Konvergenzordnung für das vereinfachte Verfahren ist nur $p = 1$.

Das Newton-Verfahren läßt sich leicht auf *Systeme nichtlinearer Gleichungen*

$$F(x) = 0$$

übertragen, wobei $F : \mathbb{R}^n \to \mathbb{R}^n$ eine stetig differenzierbare Funktion mit gewissen Zusatzeigenschaften ist. Die graphische Herleitung ist für Dimensionen $n > 1$ natürlich nicht mehr möglich. Aber die folgende Interpretation des Newton-Verfahren liefert eine einfache Verallgemeinerung. Die nichtlineare Funktion f wird in der Umgebung x_k durch das bezüglich x lineare Taylor-Polynom 1. Ordnung

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k)$$

ersetzt. Statt der nichtlinearen Gleichung $f(x) = 0$ wird dann im k-ten Schritt die lineare Ersatzgleichung $f(x_k) + f'(x_k)(x - x_k) = 0$ gelöst und deren Lösung als neue Iterierte x_{k+1} genommen. Überträgt man diese Interpretation auf den mehrdimensionalen Fall von Systemen nichtlinearer Gleichungen $F(x) = 0$ müssen wir lediglich die nichtlineare Funktion F lokal durch eine lineare Funktion ersetzen. Die Taylor-Entwicklung 1. Ordnung von F um einen Startwert x^0 ergibt

$$0 = F(x) \approx F(x^0) + F'(x^0)(x - x^0) := \overline{F}(x) \quad \text{für} \quad x \to x^0\,.$$

Die Nullstelle x^1 der linearen Ersatzabbildung $\overline{F}$ ist gerade

$$x^1 = x^0 - F'(x^0)^{-1} F(x^0),$$

falls die Jacobimatrix $F'(x^0)$ an der Stelle x^0 invertierbar ist. Dies inspiriert die *Newton-Iteration* $(k = 0, 1, \dots)$

$$F'(x^k)\Delta x^k = -F(x^k) \quad \text{mit} \quad x^{k+1} = x^k + \Delta x^k .$$

Natürlich wird nicht etwa die Inverse $F'(x^k)^{-1}$ in jedem Iterationsschritt berechnet, sondern nur die sogenannte *Newton-Korrektur* Δx^k als Lösung des obigen linearen Gleichungssystems. Wir haben also die numerische Lösung eines nichtlinearen Gleichungssystems zurückgeführt auf die numerische Lösung einer *Folge von linearen Gleichungssystemen.*

Algorithmus: Newton-Verfahren zur Lösung von $F(x) = 0$
wähle Startvektor $x^0 \in \mathbb{R}^n$
setzte $k = 0$
wiederhole
berechne Jacobi-Matrix $B_k = F'(x^k) = \frac{\partial F_i(x^k)}{\partial x^j}$
löse das lineare Gleichungssystem $B_k s^k = -F(x^k)$
setze $x^{k+1} = x^k + s^k$
setze $k = k + 1$
bis Konvergenz

Tabelle 3.10: Newton-Verfahren zur Lösung von $F(x) = 0$

Beispiel 3.28 *(Dieses Beispiel ist dem Buch Numerische Verfahren für Naturwissenschaftler und Ingenieure von H. Schwetlick und H. Kretzschmar, Fachbuchverlag Leipzig, 1991 entnommen) Bei der näherungsweisen Lösung der nichtlinearen Randwertaufgabe*

$$y'' = e^y, \quad 0 < y < 1, \quad y(0) = 0, \quad y(1) = 1$$

nach der Methode der finiten Differenzen mit der Schrittweite $h = 1/N$ entsteht das nichtlineare Gleichungssystem

$$-y_{i-1} + 2y_i - y_{i+1} + h^2 e^{y_i} = 0$$

für $i = 1, 2, \dots, N-1$, $y_0 = 0$ und $y_N = 1$. Hierbei sei $y_i \approx y(t_i)$ (y übernimmt hier die Rolle des bisherigen x). Wir haben also $F_i(y) = -y_{i-1} + 2y_i - y_{i+1} + h^2 e^{y_i} = 0$. Damit ergibt sich für die Jacobi-Matrix

$$F'(y) = \begin{pmatrix} 2 + h^2 e^{y_1} & -1 & & & \\ -1 & 2 + h^2 e^{y_2} & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 + h^2 e^{y_{N_1}} & -1 \\ & & & -1 & 2 + h^2 e^{y_N} \end{pmatrix}.$$

Diese Matrix ist diagonaldominant. Das Gleichungssystem $F'(y^k)s^k = -F(y^k)$ kann dann billig mit einer der speziellen Struktur der Problems angepassten Varainte des Gauß-Verfahrens gelöst werden. Für $N = 5$ und den Startvektor $y^0 = [0.2 \quad 0.4 \quad 0.6 \quad 0.8]^T$ ergeben sich die in Tabelle 3.11 zusammengefaßten Resultate.

k	y_1^k	y_2^k	y_3^k	y_4^k
0	0.2	0.4	0.6	0.8
1	0.095541225	0.234835111	0.423946103	0.673110201
2	0.094476803	0.232916928	0.421848024	0.671770057
3	0.094476679	0.232916700	0.421847773	0.671769903

Tabelle 3.11: Newton-Verfahren

Wenn man statt $F'(x^k)$ in jedem Iterationsschritt $F'(x^0)$ verwendet, spricht man vom vereinfachten Newton-Verfahren. Die Jacobi-Matrix wird also einmal aufgestellt. Berechnet man ihre LR-Zerlegung, so kann das in jedem Iterationsschritt auftretendende Gleichungssystem dann leicht durch Rückwärts- und Vorwärtseinsetzen gelöst werden. Die Konvergenz ist nur linear, dafür ist der Aufwand deutlicher geringer als beim Newton-Verfahren. Ein Kompromiß sind die p-stufigen Newton-Verfahren, bei denen die Jacobi-Matrix alle p Schritte neu berechnet wird.

Der Vorteil der schnellen quadratischem Konvergenz des Newton-Verfahrens wird mit einem relativen hohen Aufwand pro Schritt (Berechnung der Jacobi-Matrix und der Lösung eines linearen Gleichungssystems) erkauft. Wenn die Berechnung der partiellen Ableitungen $\partial F_i / \partial x_j$ sehr aufwendig oder aus anderen Gründen nicht zweckmäßig ist, kann man versuchen, statt mit der Jacobi-Matrix mit einer Approximation an diese zu arbeiten. Der einfachste Weg zur Approximation von $F'(x^k)$ besteht darin, die partiellen Ableitungen $\partial F_i / \partial x_j$ durch Differenzenquotienten bzgl. x_j zu ersetzen, also

$$\frac{\partial F_i}{\partial x_j} \approx \frac{F_i(x_1^k, \ldots, x_{j-1}^k, x_j^k + h_j^k, x_{j+1}^k, \ldots, x_n^k) - F_i(x_1^k, \ldots, x_{j-1}^k, x_j^k, x_{j+1}^k, \ldots, x_n^k)}{h_j^k}.$$

Man spricht dann von einem diskretisierten Newton-Verfahren. Sofern die Rechengenauigkeit genügend hoch ist, ist das Konvergenzverhalten nicht wesentlich von dem des Newton-Verfahrens verschieden. Vom Aufwand pro Schritt ist das diskretisierte Newton-Verfahren mit dem Newton-Verfahren vergleichbar.

Mit einem geringeren Aufwand pro Schritt kommen die sogenannten Quasi-Newton-Verfahren aus. Bei diesen wird $B_{k+1} = F'(x^{k+1})$ aus $B_k = F'(x^k)$ nur unter Verwendung von Δx^k und $y^k = F(x^{k+1}) - F(x^k)$ so bestimmt, daß die Quasi-Newton-Gleichung

$$B_{k+1}\Delta x^k = y^k$$

erfüllt sind. Durch die n skalaren Gleichungen sind die n^2 Elemente von B_{k+1} nicht eindeutig bestimmt (für $n > 1$). Fordert man zusätzlich z.B. noch

$$B_{k+1}r = B_k r$$

für alle r mit $r\Delta x^k = 0$ ist B_{k+1} im Falle $\Delta x^k \neq 0$ eindeutig durch die Vorschrift

$$B_{k+1} = B_k + \frac{(y^k - B_k \Delta x^k)(\Delta x^k)^T}{(\Delta x^k)^T \Delta x^k}$$

bestimmt; dies ist das Verfahren von Broyden. Pro Schritt ist nur eine Auswertung von F erforderlich, das Verfahren ist daher wesentlich billiger als das Newton-Verfahren, praktisch ist die Konvergenz häufig nur wenig langsamer als beim Newton-Verfahren.
Bei schlechtem Startwert konvergiert das Newton-Verfahren häufig schlecht oder divergiert. Um globale Konvergenz zu erzielen, d.h. Konvergenz für (fast) alle Startwerte, werden sogenannte Globalisierungstechniken eingesetzt. Eine Möglichkeit besteht in der Einführung eines Suchparameters $\alpha_k > 0$

$$x^{k+1} = x^k + \alpha_k \Delta x^k.$$

Dieser wird dann so festgelegt, daß

$$||F(x^{k+1})||_2 < ||F(x^k)||_2$$

gilt. Eine alternative Globalisierungsmöglichkeit stellen Homotopieverfahren dar.

3.4 Nichtlineare Ausgleichsprobleme

Hier betrachten wir kurz Verfahren zur Lösung eines überbestimmten Systems von N nichtlinearen Gleichungen zur Bestimmung von $n < N$ Unbekannten $x_1, x_2, \ldots, x_n$ aus N Meßdaten (z_k, y_k)

$$f_i(z_i; x_1, \ldots, x_n) = y_i.$$

Typischerweise kann ein solches Gleichungssystem nicht exakt gelöst werden. Man versucht stattdessen, die $x_1, \ldots, x_n$, so zu bestimmen, daß für

$$r_i = f_i(z_i; x_1, \ldots, x_n) - y_i, \tag{3.20}$$

$r = [r_1, \ldots, r_N]^T$ und $x = [x_1, \ldots, x_n]^T$

$$F(x) = r^T r$$

minimal wird (siehe Methode der kleinsten Quadrate).
Die notwendige Bedingung zur Minimierung der Funktion F ist für $j = 1, \ldots, n$

$$\begin{aligned} 0 &= \frac{\partial F(x)}{\partial x_j} \\ &= \sum_{i=1}^{N} (f_i(z_i; x_1, \ldots, x_n) - y_i) \frac{\partial f_i(z_i; x_1, \ldots, x_n)}{\partial x_j}, \end{aligned}$$

ein System von n nichtlinearen Gleichungen für die Unbekannten x_j. Ein solches System kann i.a. nur iterativ gelöst werden.
Daher führt man das Problem durch Linearisierung der Fehlergleichungen (3.20) auf eine Folge von linearen Ausgleichsproblemen zurück. Ausgehend von einem Startvektor $x^{(0)} = [x_1^{(0)}, \ldots, x_n^{(0)}]^T$ bestimmt man weitere Näherungslösungen $x^{(1)}, x^{(2)}, \ldots$ wie folgt: Für $x^{(i)}$ berechne man die Lösung $s^{(i)}$ des linearen Ausgleichsproblems

$$\min_{s \in \mathbb{R}^N} ||r(x^{(i)}) - J(x^{(i)})s||_2^2 \tag{3.21}$$

mit der Jacobi-Matrix

$$J(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x) & \cdots & \frac{\partial f_1}{\partial x_n}(x) \\ \vdots & & \vdots \\ \frac{\partial f_N}{\partial x_1}(x) & \cdots & \frac{\partial f_N}{\partial x_n}(x) \end{bmatrix}$$

und

$$r(x) = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} - \begin{bmatrix} f_1(z_1; x_1, \ldots, x_n) \\ \vdots \\ f_N(z_N; x_1, \ldots, x_n) \end{bmatrix}.$$

Dies kann wie bei der Methode der kleinsten Quadrate beschrieben, mittels der QR-Zerlegung von J geschehen. Der Vektor

$$x^{(i+1)} = x^{(i)} + s^{(i)}$$

ist dann im günstigsten Fall eine bessere Näherung an die gesuchte Lösung.
Das Iterationsverfahren bezeichnet man als Gauß-Newton-Methode, da die Korrektur $s^{(i)}$ nach der von Gauß verwendeten Methode der kleinsten Quadrate ermittelt wurde und sich die linearisierten Fehlergleichungen im Sonderfall $N = n$ auf die linearen Gleichungen reduzieren, die in der Methode von Newton zur Lösung von nichtlinearen Gleichungen auftreten.
Die mit der Gauß-Newton-Methode gebildete Folge von Vektoren $x^{(k)}$ braucht bei ungeeigneter Wahl des Startvektors oder bei kritischen Ausgleichsproblemen nicht gegen die gesuchte Lösung des nichtlinearen Problems zu konvergieren. Um stets eine gegen die Lösung konvergierende Folge von Vektoren $x^{(k)}$ zu konstruieren, sollte

$$F(x^{(k+1)}) < F(x^{(k)}) \tag{3.22}$$

erfüllt sein. Dazu muß in jedem Schritt eine Abstiegsrichtung $v^{(k)}$ bekannt sein, so daß mit

$$x^{(k+1)} = x^{(k)} + tv^{(k)}, \qquad t > 0$$

die Bedingung (3.22) erfüllt ist. Eine Abstiegsrichtung ist der negative Gradient der Funktion $F(x)$ im Punkt $x^{(k)}$ dar. Diese Richtung ist mittels

$$v^{(k)} = -(J^{(k)})^T r^{(k)}$$

berechenbar, wobei $J^{(k)}$ die Jacobi-Matrix und $r^{(k)}$ den Residuenvektor für $x^{(k)}$ bezeichnen. Wird der Parameter t so bestimmt, daß

$$F(x^{(k+1)}) = \min_t F(x^{(k)} + tv^{(k)})$$

gilt, spricht man von der Methode des steilsten Abstiegs, welche in der Regel sehr langsam gegen die gesuchte Lösung konvergiert.
Eine andere, besser geeignete Abstiegsrichtung ist der bei der Gauß-Newton-Methode berechnete Korrekturvektor $s^{(k)}$. Der Vektor

$$x^{(i+1)} = x^{(i)} + 2^{-k} s^{(i)}, \quad k = 0, 1, 2, 3, \ldots$$

für welchen zum ersten Mal $F(x^{(i+1)}) < F(x^{(i)})$ gilt, ist dann eine bessere Näherung an die gesuchte Lösung. Bei ungünstiger Wahl des Startvektors $x^{(0)}$ kann die Konvergenz der Folge $x^{(k)}$ zu Beginn der Iteration sehr langsam sein. In der Nähe des Lösungsvektors x ist die Konvergenz annährend quadratisch.
Eine effizientere Methode stammt von *Marquardt*. Um eine günstigere Abstiegsrichtung zu bestimmen, betrachtet er die Aufgabe, mit $C = J^{(k)}$ und $d = r^{(k)}$ den Vektor v als Lösung des Extremalproblems

$$||Cv + d||_2^2 + \lambda^2 ||v||_2^2 = \text{ min}, \ \lambda > 0 \tag{3.23}$$

zu bestimmen. Bei gegebenem Wert der Parameters λ ist v die Lösung des Systems von Fehlergleichungen nach der Methode der kleinsten Quadrate

$$\tilde{C}v + \tilde{d} = \tilde{\rho} \quad \text{mit} \quad \tilde{C} := \begin{pmatrix} C \\ \lambda I \end{pmatrix} \in \mathbb{R}^{(N+n)\times n}, \tilde{d} := \begin{pmatrix} d \\ 0 \end{pmatrix} \in \mathbb{R}^{N+n}, \tilde{\rho} \in \mathbb{R}^{N+n} \tag{3.24}$$

Für jedes $\lambda > 0$ hat die Matrix $\tilde{C}$ den Maximalrang n unabhängig vom Rang von C. Im Vergleich zur Gauß-Newton Methode wurde jenes Fehlergleichungssystem (3.21) um n Gleichungen erweitert und auf diese Weise regularisiert. Der Lösungsvektor v besitzt folgende Eigenschaften:

1. Der Vektor $v = v^{(k+1)}$ ist eine Abstiegsrichtung (solange $\nabla F(x^{(k)}) \neq 0$ ist).

2. Die euklidische Norm $||v||_2$ des Vektors ist mit zunehmendem λ eine monoton abnehmende Funktion.

Im Verfahren von Marquardt zur Minimierung der Summe der Quadrate der Residuen wird die euklidische Norm des Vektors $v^{(k+1)}$ durch den Parameter λ so gesteuert, daß mit

$$x^{(k+1)} = x^{(k)} + v^{(k+1)}, \quad F(x^{(k+1)}) < F(x^{(k)}), \qquad k = 0, 1, 2, \ldots \tag{3.25}$$

gilt. Die Wahl des Wertes λ erfolgt auf Grund des Rechenablaufes. Ist die Bedingung (3.25) für das momentane λ erfüllt, dann soll λ für den nachfolgenden Schritt verkleinert werden, beispielsweise halbiert. Ist aber (3.25) für das momentane λ nicht erfüllt, soll λ solange vergrößert, beispielsweise verdoppelt werden, bis ein Vektor $v^{(k+1)}$ resultiert, für den die Bedingung gilt. Selbstverständlich muß mit dem Startvektor $x^{(0)}$ auch ein Startwert $\lambda^{(0)}$ vorgegeben werden. Ein problemabhängiger Vorschlag ist

$$\lambda^{(0)} = ||C^{(0)}||_F / \sqrt{nN} = \sqrt{\frac{1}{nN} \sum_{i,j} (c_{ij}^{(0)})^2}$$

mit der Frobenius-Norm der Matrix $C^{(0)}$ zum Startvektor $x^{(0)}$.
Ein Iterationsschritt des Verfahrens von Marquardt erfordert die Berechnung des Vektors $v^{(k+1)}$ aus dem Fehlergleichungssystem (3.24) für möglicherweise mehrere Werte des Parameters λ. Um diesen Schritt möglichst effizient zu gestalten, erfolgt die Behandlung von (3.24) in zwei Teilen. In einem vorbereitenden Teilschritt werden die ersten N von λ unabhängigen Fehlergleichungen mit einer orthogonalen Matrix Q_1 so transformiert, daß

$$Q_1^T \tilde{C} = \begin{pmatrix} R_1 \\ 0_1 \\ \lambda I \end{pmatrix}, \qquad Q_1^T \tilde{d} = \begin{pmatrix} \hat{d}_1 \\ \hat{d}_2 \\ 0 \end{pmatrix}, \qquad R_1 \in \mathbb{R}^{n\times n}, 0_1 \in \mathbb{R}^{(N-n)\times n} \tag{3.26}$$

gilt. Unter der Annahme, daß C Maximalrang hat, ist R_1 eine reguläre obere Dreiecksmatrix, und die Transformation kann entweder mit der Methode von Givens oder Householder erfolgen. Auf jeden Fall bleiben die letzten n Fehlergleichungen unverändert. Ausgehend von (3.26) wird zu gegebenem λ mit einer orthogonalen Matrix Q_2 die Transformation beendet, um die Matrizen und Vektoren

$$Q_2^T Q_1^T \tilde{C} = \begin{pmatrix} R_2 \\ 0_1 \\ 0_2 \end{pmatrix}, \qquad Q_2^T Q_1^T \tilde{d} = \begin{pmatrix} \tilde{d}_1 \\ \hat{d}_2 \\ \hat{d}_3 \end{pmatrix}, \qquad R_2, 0_2 \in \mathbb{R}^{n \times n}$$

zu erhalten. Der gesuchte Vektor $v^{(k+1)}$ ergibt sich aus $R_2 v^{(k+1)} + \tilde{d}_1 = 0$ mit der stets reguären oberen Dreiecksmatrix R_2 durch den Prozeß des Rückwärtseinsetzens. Muß der Wert λ vergrößert werden, so ist nur der zweite Teilschritt zu wiederholen. Dazu sind R_1 und $\hat{d}_1$ als Ergebnis der ersten Teils abzuspeichern. Da die Nullmatrix 0_1 und der Teilvektor $\hat{d}_2$ von (3.26) für den zweiten Teilschritt bedeutungslos sind, kann er sowohl speicherökonomisch als auch sehr effizient durchgeführt werden. Die sehr spezielle Struktur der noch zu behandelnden Fehlergleichungen des zweiten Teilschrittes weist auf die Anwendung der Givens-Transformation hin.

3.5 Polynom-Interpolation

Häufig tritt die Situation auf, daß statt einer Funktion $f : \mathbb{R} \to \mathbb{R}$ nur einige diskrete Funktionswerte $f(t_i)$ und eventuell noch Ableitungen $f^{(j)}(t_i)$ an endlich vielen Punkten t_i gegeben sind. Dies trifft z.B. zu, wenn die Funktion f in Form experimenteller Daten vorliegt. Historisch trat das Problem bei der Berechnung von zusätzlichen Funktionswerten zwischen tabellierten Werten auf (es gab z.B. Tabellen für $\log, \sin, \cos, \tan.$). Heute ist eines der bedeutendsten Anwendungsfelder die Computer-Grafik. Ist man an dem gesamten Verlauf der Funktion interessiert, so sollte man aus den gegebenen Daten

$$f^{(j)}(t_i) \qquad \text{für} \qquad i = 0, \ldots, n \qquad \text{und} \qquad j = 0, \ldots, c_i$$

eine Funktion ϕ konstruieren, die sich möglichst wenig von der ursprünglichen Funktion f unterscheidet. Zudem sollte ϕ eine leicht auswertbare Funktion sein, wie z.B.

- ein Polynom $p(x) = \sum\limits_{j=0}^{n} \quad a_j x^j$
- ein trigonometrisches Polynom $p(x) = \sum\limits_{j=0}^{n} \quad a_j e^{ijx}$
- eine rationale Funktion $p(x) = \left(\sum\limits_{j=0}^{r} \quad a_j x^j \right) \left(\sum\limits_{j=0}^{s} \quad b_j x^j \right)^{-1}$
- eine Exponentialsumme $p(x) = \sum\limits_{j=0}^{m} \quad a_j x^{\lambda j x}$
- ...

Eine erste naheliegende Forderung an die Funktion ϕ ist die sogenannte *Interpolationseigenschaft:* ϕ soll an den Knoten (oder Stützstellen) t_i mit der Funktion f übereinstimmen

$$\phi^{(j)}(t_i) = f^{(j)}(t_i) \quad \forall i, j \,.$$

Die Werte $f^{(j)}(t_i)$ heißen auch Stützwerte.
Wir beginnen nun mit dem einfachsten Fall, daß nur die Funktionswerte

$$f_i := f(t_i) \quad \text{für} \quad i = 0, \ldots, n$$

an den paarweise verschiedenen Knoten $t_0, \ldots, t_n$ gegeben sind. Wir suchen ein Polynom p vom Grad $\leq n$,

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

mit $a_0, a_1, \ldots, a_n \in \mathbb{R}$, welches f an den $n+1$-Knoten $t_0, \ldots, t_n$ interpoliert, d.h.

$$p(t_i) = f_i \quad \text{für} \quad i = 0, \ldots, n \,.$$

Satz 3.29 *(Existenz- und Eindeutigkeitssatz):*
Zu $n+1$ Stützpunkten (t_i, f_i) für $i = 0, \ldots, n$ mit paarweise verschiedenen Knoten $t_0, \ldots, t_n$ existiert genau ein Interpolationspolynom $p \in \pi_n$, d.h.

$$p(t_i) = f_i \quad \textit{für} \quad i = 0, \ldots, n\,,$$

p wird mit $p(f \mid t_0, \ldots, t_n)$ bezeichnet.

Beweis der Eindeutigkeit:

Sind $p, q \in \pi_n$ mit $p(t_i) = q(t_i) = f_i$ für $i = 0, \ldots, n$, so ist $r := p - q \in \pi_n$, und r besitzt $n+1$ Nullstellen, nämlich $t_0, \ldots, t_n$, also ist $r \equiv 0$. $\surd$

Beweis der Existenz (konstruktiv):

Schreiben wir p in Koeffizientendarstellung

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

so lassen sich die Interpolationsbedingungen $p(t_i) = f_i$ als lineares Gleichungssystem

$$\underbrace{\begin{pmatrix} 1 & t_0 & t_0^2 & \ldots & t_0^n \\ 1 & t_1 & t_1^2 & \ldots & t_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & t_n & t_n^2 & \ldots & t_n^n \end{pmatrix}}_{V} \underbrace{\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix}}_{a} = \underbrace{\begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix}}_{f}$$

formulieren. Die Matrix V heißt *Vandermonde-Matrix.* Für die Determinante von V gilt

$$\det V = \prod_{i=0}^{n} \quad \prod_{j=i+1}^{n} \quad (t_i - t_j).$$

Sie ist genau dann von Null verschieden, wenn die Knoten $t_0, \ldots, t_n$ paarweise verschieden sind. Das Gleichungssystem $Va = f$ ist also unter den gegebenen Voraussetzungen eindeutig lösbar, $a = V^{-1} f$. $\surd$

Praktisch ist der im Beweis angegebene Lösungsweg unbrauchbar. Zum einen erfordert das Lösen des Gleichungssystems übermäßig viel Aufwand im Vergleich zu den folgenden Methoden; andererseits sind die Vandermonde-Matrizen i. a. für hohe Dimensionen n fast singulär.

Bisher haben wir das Interpolationspolynom bezüglich der monomialen Basis $\{1, t, t^2, \ldots, t^n\}$ dargestellt. Eine alternative Basis zur Darstellung des Interpolationspolynoms bilden die sogenannten *Lagrange-Polynome* $L_0, \ldots, L_n$. Sie sind definiert als die eindeutig bestimmten Interpolationspolynome L_i vom Grad $\leq n$ mit

$$L_i(t_j) = \delta_{ij}.$$

Mit etwas Nachdenken findet man rasch die zugehörige explizite Form

$$L_i(t) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{t - t_j}{t_i - t_j}. \tag{3.27}$$

Mit

$$p(t) = \sum_{i=0}^{n} \quad f_i L_i(t) \tag{3.28}$$

gilt offensichtlich

$$p(t_j) = \sum_{i=0}^{n} \quad f_i L_i(t_j) = \sum_{i=0}^{n} \quad f_i \delta_{ij} = f_j.$$

Beispiel 3.30 *Sei $n = 2$. Seien die 3 Stützpunkte*

$$(t_0, f_0) = (1, 2)\,, (t_1, f_1) = (2, 1)\,, (t_2, f_2) = (3, 2)$$

gegeben. Gesucht ist nun ein Interpolationspolynom p vom Grad 2 mit $p(t_i) = f_i$ für $i = 0, 1, 2$. Wegen (3.27), (3.28) ergibt sich

$$\begin{aligned}
L_0(t) &= \tfrac{(t-2)(t-3)}{(1-2)(1-3)} = \tfrac{1}{2}(t^2 - 5t + 6) \\
L_1(t) &= \tfrac{(t-1)(t-3)}{(2-1)(2-3)} = -(t^2 - 4t + 3) \\
L_2(t) &= \tfrac{(t-1)(t-2)}{(3-1)(3-2)} = \tfrac{1}{2}(t^2 - 3t + 2) \\
p(t) &= \sum_{j=0}^{2} \quad f_j L_j(t) = 2L_0(t) + L_1(t) + 2L_2(t) \\
&= t^2 - 5t + 6 - t^2 + 4t - 3 + t^2 - 3t + 2 \\
&= t^2 - 4t + 5\,.
\end{aligned}$$

Die Lagrange-Darstellung (3.28) ist für praktische Zwecke zu rechenaufwendig, jedoch für viele theoretische Fragestellungen vorteilhaft.

Wir wollen uns nun der zweiten Forderung an das Interpolationspolynom zuwenden, der *Approximationseigenschaft*, und analysieren, inwieweit die Polynome $p(f \mid t_0, \ldots, t_n)$ die ursprüngliche Funktion f approximieren.

Satz 3.31 *Sei $P(t)$ das Interpolationspolynom vom Grad n einer Funktion $f \in C^{n+1}[a, b]$ zu den paarweise verschiedenen Knoten $a \leq t_0, \ldots, t_n \leq b$. Dann gibt es für alle $\bar{t} \in [a, b]$ ein $\tau = \tau(\bar{t}) \in]a, b[$, so daß*

$$f(\bar{t}) - P(\bar{t}) = \frac{f^{(n+1)}(\tau)}{(n+1)!} \quad (\bar{t} - t_0) \cdots (\bar{t} - t_n).$$

Bemerkungen:

1) Der Approximationsfehler hängt offenbar entscheidend von der Wahl der Knoten $t_0, \ldots, t_n$ in Form von
$$w_{n+1}(t) = (t - t_0) \cdots (t - t_n)$$
ab. Man kann zeigen, daß sich w_{n+1} für paarweise verschiedene Knoten auf einem Intervall $[a, b]$ minimieren läßt
$$\max_{t\in[a,b]} \mid w_{n+1}(t) \mid = \min\,,$$
indem man für die Knoten t_i die Nullstellen der *Tschebyscheff- Polynome* wählt, d.h.
$$t_i = \cos\left(\frac{2i+1}{2n+2}\pi\right) \quad i = 0, \ldots, n\,.$$

2) Für lediglich stetige Funktionen $f \in C[a, b]$ und die Supremumsnorm $\|f\| = \sup\limits_{t\in[a,b]} f(t)$ kann der Approximationsfehler prinzipiell über alle Schranken wachsen. Genauer besagt der Satz von Faber: Zu jeder Folge $\{T_k\}$ von Stützstellensätzen $T_k = \{t_{k0}, \ldots, t_{kn_k}\} \subset [a, b]$ existiert eine stetige Funktion $f \in C[a, b]$, so daß die Folge $\{p_k\}$ der zu den T_k gehörenden Interpolationspolynome nicht gleichmäßig gegen f konvergiert.

3.5.1 Algorithmus von Aitken-Neville

Ist man an dem Interpolationspolynom p nur an einer Stelle t interessiert, so erweist sich die rekursive Berechnung von $p(t)$ als effektivste Methode. Sie geht von folgender einfacher Beobachtung aus, dem sogenannten *Lemma von Aitken.*

Lemma 3.32 *Für das Interpolationspolynom $p = p(f \mid t_0, \ldots, t_n)$ gilt die Rekursionsformel*
$$p(f \mid t_0, \ldots, t_n) = \frac{(t_0 - t)p(f \mid t_1, \ldots, t_n) - (t_n - t)p(f \mid t_0, \ldots, t_{n-1})}{t_0 - t_n}\,. \tag{3.29}$$

Beweis: Sei $\varphi(t)$ definiert als der Ausdruck auf der rechten Seite von (3.29). Dann ist φ ein Polynom vom Grad $\leq n$ und
$$\varphi(t_i) = \frac{(t_0 - t_i)f_i - (t_n - t_i)f_i}{t_0 - t_n} = f_i \quad \text{für} \quad i = 1, \ldots, n-1\,.$$
Ebenso leicht folgt $\varphi(t_0) = f_0$ und $\varphi(t_n) = f_n$ und daher die Behauptung. $\surd$

Die Interpolationspolynome für nur einen Stützpunkt sind nichts weiter als die Konstanten
$$p(f|t_i) = f_i \quad \text{für} \quad i = 0, \ldots, n\,.$$

Vereinfachen wir die Notation für festes t durch

$$p_{ik} := p(f|t_{i-k},\dots,t_i)(t) \quad \text{für} \quad i \geq k\,,$$

so läßt sich der Wert $p_{nn} = p(f|t_0,\dots,t_n)(t)$ nach dem *Schema von Neville* berechnen

$$\begin{array}{ccccccc}
p_{00} & & & & & & \\
 & \searrow & & & & & \\
p_{10} & \rightarrow & p_{11} & & & & \\
\vdots & & & \ddots & & & \\
p_{n-1,0} & \rightarrow & \cdots & \rightarrow & p_{n-1,n-1} & & \\
 & & & & & \searrow & \\
p_{n0} & \rightarrow & \cdots & \rightarrow & p_{n,n-1} & \rightarrow & p_{nn}
\end{array}$$

gemäß der Vorschrift

$$\begin{array}{lcll}
p_{i0} & = & f_i & i = 0,\dots,n \\
p_{ik} & = & p_{i,k-1} + \frac{t-t_i}{t_i-t_{i-k}}\,(p_{i,k-1} - p_{i-1,k-1}) & i \geq k\,.
\end{array}$$

Beispiel 3.33 *Berechnung von* $\sin 62^\circ$ *aus den Stützpunkten*

$$(50^\circ, \sin 50^\circ), (55^\circ, \sin 55^\circ), \dots, (70^\circ, \sin 70^\circ)$$

mit dem Aitken-Neville-Algorithmus.

t_i	$\sin t_i$				
50°	0.7660444				
55°	0.8191520	$0.\underline{8}935027$			
60°	0.8660254	$0.\underline{8}847748$	$0.\underline{88}30292$		
65°	0.9063078	$0.\underline{8}821384$	$0.\underline{8829}293$	$0.\underline{88294}93$	
70°	0.9396926	$0.\underline{8}862768$	$0.\underline{8829}661$	$0.\underline{88294}65$	$0.\underline{8829476}$

3.5.2 Newtonsche dividierte Differenzen

Die rekursive Struktur der Interpolationspolynome nach Lemma 3.32 läßt sich auch zur Bestimmung des gesamten Polynoms $P(f \mid t_0,\dots,t_n)$ nutzen. Dies gilt auch für die verallgemeinerte Interpolationsaufgabe, bei der neben den Funktionswerten $f(t_i)$ noch Ableitungen an den Knoten gegeben sind, der sogenannten *Hermite-Interpolation*. Dazu führen wir folgende praktische Notation ein. Wir lassen zu, daß in der Folge $\Delta := \{t_i\}_{i=0,\dots,n}$ mit

$$a = t_0 \leq t_1 \leq \cdots \leq t_n = b$$

Knoten mehrfach auftreten. Sind an einem Knoten t_i der Funktionswert $f(t_i)$ und die Ableitungen $f'(t_i),\dots,f^{(k)}(t_i)$ bis zu einem Grad k gegeben, so soll t_i in der Folge Δ gerade $(k+1)$-mal auftauchen. Die gleichen Knoten numerieren wir mit

$$d_i := \max\{j \mid t_i = t_{i-j}\}$$

von links nach rechts durch, z.B.

t_i	t_0	$=$	t_1	$<$	t_2	$=$	t_3	$=$	t_4	$<$	t_5	$<$	t_6	$=$	t_7
d_i	0		1		0		1		2		0		0		1

.

Definieren wir nun für $i = 0, \ldots, n$ lineare Abbildungen

$$\mu_i : C^n[a,b] \to \mathbb{R}, \quad \mu_i(g) := g^{(d_i)}(t_i)\,,$$

so lautet die Aufgabe der *Hermite-Interpolation*: Suche ein Polynom p vom Grad $\leq n$, so daß

$$\mu_i(p) = \mu_i(f) \quad \text{für alle} \quad i = 0, \ldots, n\,. \tag{3.30}$$

Die Lösung $p = p(f|t_0, \ldots, t_n) \in \pi_n$ der Interpolationsaufgabe (3.30) heißt *Hermite-Interpolierende* von f an den Knoten $t_0, \ldots, t_n$ (π_n = Menge aller Polynome vom Grad $\leq n$). Die Existenz und Eindeutigkeit folgt wie in Satz 3.29.

Satz 3.34 *Zu jeder Funktion $f \in C^n[a,b]$ und jeder monotonen Folge*

$$a = t_0 \leq t_1 \leq \cdots \leq t_n = b$$

von (nicht notwendig verschiedenen) Knoten gibt es genau ein Polynom P vom Grad $\leq n$, so daß

$$\mu_i(p) = \mu_i(f) \quad \textit{für alle} \quad i = 0, \ldots, n\,.$$

Sind alle Knoten paarweise verschieden, so erhalten wir die Lagrange-Interpolierende

$$P(f \mid t_0, \ldots, t_n) = \sum_{i=0}^{n} f(t_i) L_i$$

zurück. Stimmen alle Knoten überein, $t_0 = t_1 = \cdots = t_n$, so ist das Interpolationspolynom die abgebrochene Taylor-Reihe um $t = t_0$,

$$P(f \mid t_0, \ldots, t_n)(t) = \sum_{j=0}^{n} \frac{(t-t_0)^j}{j!} f^{(j)}(t_0),$$

auch *Taylor-Interpolation* genannt.

Zur Darstellung des Interpolationspolynoms verwenden wir die sogenannte *Newton-Basis* $\omega_0, \ldots, \omega_n$ des Polynomraums π_n:

$$\omega_i(t) := \prod_{j=0}^{i-1} (t - t_j)\,, \quad \omega_i \in \pi_i\,.$$

$$p(x) = \sum_{j=0}^{n} a_j \omega_j(t)$$

ist dann das Hermite-Interpolationspolynom vom Grad n. Die Koeffizienten a_j bezüglich der Newton-Basis sind die sogenannten *dividierten Differenzen.* Um die Abhängigkeit der a_j von f und den Stützstellen $x_0, \dots, x_n$ zu verdeutlichen, schreibt man auch

$$a_j = [x_0, \dots, x_j]f$$

für die $j$te dividierte Differenz.

Beispiel 3.35 *Gegeben seien die Stützwerte* $(x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3)$. *Gesucht ist ein Polynom*

$$p(x) = a_1 + a_2(x - x_0) + a_3(x - x_0)(x - x_1) + a_4(x - x_0)(x - x_1)(x - x_2),$$

für welches $p(x_i) = y_i$ *für* $i = 0, 1, 2, 3$ *gilt.*
Es folgt

$$\begin{aligned}
p(x_0) &= y_0 = a_1 \\
p(x_1) &= y_1 = a_1 + a_2(x_1 - x_0) \\
p(x_2) &= y_2 = a_1 + a_2(x_2 - x_0) + a_3(x_2 - x_0)(x_2 - x_1) \\
p(x_3) &= y_3 = a_1 + a_2(x_3 - x_0) + a_3(x_3 - x_0)(x_3 - x_1) \\
&\quad + a_4(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)(x_3 - x_4)
\end{aligned}$$

und

$$\begin{aligned}
a_1 &= y_0 \\
a_2 &= \frac{y_1 - a_1}{x_1 - x_0} \\
a_3 &= \frac{y_2 - (a_1 + a_2(x_2 - x_0))}{(x_2 - x_0)(x_2 - x_1)} \\
a_4 &= \frac{y_3 - (a_1 + a_2(x_3 - x_0) + a_3(x_3 - x_0)(x_3 - x_1)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)}
\end{aligned}$$

Man kann zeigen:

Lemma 3.36 *Für die dividierten Differenzen* $[t_0, \dots, t_n]f$ *gilt* ($f \in C^n[a, b]$)*:*

i) $[t_0, \dots, t_n]P = 0$ *für alle* $P \in \pi_{n-1}$.

ii) Für zusammenfallende Knoten $t_0 = \dots = t_n$ *gilt*

$$[t_0, \dots, t_n]f = f^{(n)}(t_0)/n!. \tag{3.31}$$

iii) Für $t_i \neq t_j$ *gilt die Rekursionsformel*

$$[t_0, \dots, t_n]f = \frac{[t_0, \dots, \hat{t_i}, \dots, t_n]f - [t_0, \dots, \hat{t_j}, \dots, t_n]f}{t_j - t_i} \tag{3.32}$$

Durch die Eigenschaften ii) und iii) lassen sich die dividierten Differenzen rekursiv aus den Funktionswerten und Ableitungen $f^{(j)}(t_i)$ von f an den Knoten t_i berechnen.
Es gilt

Korollar 3.37 *Falls $f \in C^{n+1}[a,b]$, so gilt für den Approximationsfehler der Hermite-Interpolierenden $P(f \mid t_0, \dots, t_n)$ und $t_i, t \in [a,b]$, daß*

$$f(t) - P(f \mid t_0, \dots, t_n)(t) = \frac{f^{(n+1)}(\tau)}{(n+1)!}\, \omega_{n+1}(t)$$

für ein $\tau = \tau(t) \in]a,b[$.

Für paarweise verschiedene Knoten $t_0 < \dots < t_n$ lassen sich die dividierten Differenzen aufgrund der Rekursionsformel (3.32) ähnlich wie beim Schema von Neville anordnen:

$$\begin{array}{lclcccccc}
f_0 & = & [t_0]f & & & & & & \\
 & & & \searrow & & & & & \\
f_1 & = & [t_1]f & \to & [t_0,t_1]f & & & & \\
\vdots & & & & & \ddots & & & \\
f_{n-1} & = & [t_{n-1}]f & \to & \cdots & \to & [t_0,\dots,t_{n-1}]f & & \\
 & & & & & & & \searrow & \\
f_n & = & [t_n]f & \to & \cdots & \to & [t_1,\dots,t_n]f & \to & [t_0,\dots,t_n]f
\end{array}$$

Beispiel 3.38 *Wir berechnen das Interpolationspolynom zu den Werten*

t_i	*0*	*1*	*2*	*3*
f_i	*1*	*2*	*0*	*1*

mit Hilfe der Newtonschen dividierten Differenzen.

$$\begin{array}{llll}
[t_0]f = 1 & & & \\
[t_1]f = 2 & [t_0,t_1]f = 1 & & \\
[t_2]f = 0 & [t_1,t_2]f = -2 & [t_0,t_1,t_2]f = -3/2 & \\
[t_3]f = 1 & [t_2,t_3]f = 1 & [t_1,t_2,t_3]f = 3/2 & [t_0,t_1,t_2,t_3]f = 1,
\end{array}$$

also

$$(\alpha_0, \alpha_1, \alpha_2, \alpha_3) = (1, 1, -3/2, 1)\,.$$

Das Interpolationspolynom ist daher

$$\begin{aligned}
P(t) &= 1 + 1(t-0) + (-3/2)(t-0)(t-1) + 1(t-0)(t-1)(t-2) \\
&= t^3 - 4.5t^2 + 4.5t + 1\,.
\end{aligned}$$

Eine weitere wichtige Eigenschaft dividierter Differenzen ist die *Leibniz-Regel*: Seien $g, h \in C^n$ und $t_0 \leq t_1 \leq \cdots \leq t_n$ eine beliebige Knotenfolge. Dann gilt

$$[t_0, \ldots, t_n]gh = \sum_{i=0}^{n} [t_0, \ldots, t_i]g[t_i, \ldots, t_n]h$$

Beispiel 3.39 *Um den Einfluß der Interpolationsordnung n auf den Interpolationsfehler zu demonstrieren, betrachte die Funktion (Beispiel von Runge)*

$$f(x) = \frac{1}{1 + x^2}.$$

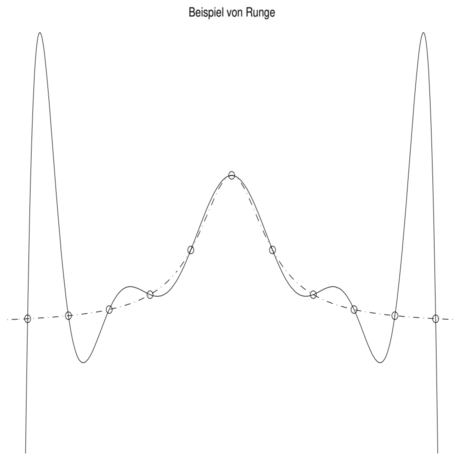


Abbildung 3.9: Beispiel von Runge

In Abbildung 3.9 ist der Verlauf dieser Funktion im Intervall [-5.5, 5.5] angegeben. Dazu ist das Interpolationspolynom p_{10} abgebildet, die mittels o gekennzeichneten Werte wurden als Stützstellen vorgegeben. Man sieht, daß das Interpolationspolynom insbesondere zwischen den äußeren Stützstellen zu starken Oszillationen neigt.

Ein kleiner Fehler läßt sich mit äquidistanten Stützstellen und höheren Polynomgraden meist nicht erreichen.

3.5.3 Spline Interpolation

Wir haben gesehen, daß die Polynominterpolation nicht unbedingt gute Ergebnisse liefert, insbesondere daß der Fehler stark oszillieren kann. Als Ausweg kann man die Interpolation durch stückweise Polynome durchführen. Dies ist ursprünglich schon im Schiffbau ein übliches Verfahren, daher auch der Name.
Heute sind Splines die wichtigsten Werkzeuge bei der Lösung von Randwertproblemen für gewöhnliche und partielle Differentialgleichungen, bei der Computergraphik, und vielen anderen Anwendungen, weil sie sehr schön glatte Kurven liefern.

Sei $\Delta := \{t_0 < t_1 < \dots t_n = t_0 + a\}$ eine Unterteilung des Intervalls $[t_0, t_0 + a]$. Eine zu Δ gehörige Spline-Funktion S_Δ ist eine reelle Funktion S_Δ mit den folgenden Eigenschaften:

a) S_Δ ist auf $[t_0, t_0 + a]$ $k-$mal stetig differenzierbar.

b) Auf jedem Teilintervall $[t_i, t_{i+1}]$ $i = 0, \dots, n-1$ stimmt S_Δ mit einem Polynom $k+1$-ten Grades überein.

Wir werden hier beispielhaft nur kubische Splines betrachten, d.h. stückweise Polynome 3.Grades. Dazu brauchen wir natürlich Interpolationsbedingungen:

$$S_\Delta(t_i) = y_i \quad i = 0, \dots, n. \tag{3.33}$$

Nun fordern wir, daß S_Δ auf dem Intervall (t_i, t_{i+1}) für $i = 0, \dots, n-1$ ein Polynom 3. Grades ist:

$$S_{\Delta,i}(t) = \alpha_i + \beta_i(t - t_i) + \gamma_i(t - t_i)^2 + \delta_i(t - t_i)^3.$$

Wir haben also 4 Parameter pro Intervall zu bestimmen, bei n Intervallen sind das $4n$ Parameter:

$$(\alpha_i, \beta_i, \gamma_i, \delta_i); \quad i = 0, \dots, n-1$$

Als Bedingungen haben wir $2n$ Bedingungen in (3.33). Dann soll die Funktion insgesamt 2 mal stetig differenzierbar sein, d.h. insgesamt haben wir

$$\left.\begin{array}{lcll} S'_{\Delta,i}(t_{i+1}) & = & S'_{\Delta,i+1}(t_{i+1}) & i = 0, \dots, n-2 \\ S''_{\Delta,i}(t_{i+1}) & = & S''_{\Delta,i+1}(t_{i+1}) & i = 0, \dots, n-2 \\ S_{\Delta,i}(t_i) & = & y_i & i = 0, \dots, n-1 \\ S_{\Delta,i}(t_{i+1}) & = & y_{i+1} & i = 0, \dots, n-1 \end{array}\right\} \begin{array}{c} n-1 \\ n-1 \\ n \\ n \end{array} \tag{3.34}$$

also $4n - 2$ Bedingungen. Das reicht nicht aus, um alle Parameter zu bestimmen, daher wählt man zusätzlich die folgenden Bedingungen
a) natürlicher Spline

$$S''_{\Delta,0}(t_0) = S''_{\Delta,n-1}(t_n) = 0 \tag{3.35}$$

b) periodischer Spline, falls die Werte y_i periodisch sind

$$S^{(i)}_{\Delta,0}(t_0) = S^{(i)}_{\Delta,n-1}(t_n) \quad i = 0, 1, 2 \tag{3.36}$$

c)

$$S'_{\Delta,0}(t_0) = y'_0, \quad S'_{\Delta,n-1}(t_n) = y'_n \quad \text{für gegebene} \quad y'_0, y'_n. \tag{3.37}$$

Satz 3.40 *Gegeben sei $\Delta := \{t_0 < \dots < t_n = t_0 + a\}, y = \{y_0, \dots, y_n\}$, so gibt es einen* eindeutigen *Spline S_Δ mit S_Δ eingeschränkt auf $[t_i, t_{i+1}]$ Polynom 3. Grades. Das heißt S_Δ ist zweimal stetig differenzierbar und S_Δ erfüllt (3.33) und entweder (3.35) oder falls $y_0 = y_n$ (3.36).*

Eine wichtige Eigenschaft, die Splines für Anwendungen besonders interessant macht, ist: Die Splinefunktion ist unter allen auf $[t_0, t_0 + a]$ zweimal stetig-differenzierbaren Funktionen mit $f(t_i) = y_i$ die Funktion mit der *geringsten Krümmung*, stellt also sozusagen die glatteste Funktion dar, die diese Punkte interpoliert. Daher werden heute in der Industrie die Oberflächen (von Autos, Flugzeugen, Schiffen) durch (mehrdimensionale) Splines dargestellt.
Zur Berechnung der Splines führen wir Momente M_i ein:

$$M_i := S''_{\Delta,i}(t_i) \quad i = 0, \dots, n,$$

d.h. die Werte der 2. Ableitung von S_Δ an den Stützstellen. Da S_Δ ein stückweise Polynom 3. Grades war, ist S''_Δ eine stückweise lineare Funktion und wegen der Bedingungen (3.34) gilt (2-Punkte-Form der Geradengleichung):

$$S''_\Delta(t) = M_i \frac{t_{i+1} - t}{t_{i+1} - t_i} + M_{i+1} \frac{t - t_i}{t_{i+1} - t_i} \quad \text{für} \quad t \in [t_i, t_{i+1}].$$

Durch Integration erhalten wir $S'_\Delta(t)$ und S_Δ

$$S'_\Delta(t) = -M_i \frac{(t_{i+1} - t)^2}{2(t_{i+1} - t_i)} + M_{i+1} \frac{(t - t_i)^2}{2(t_{i+1} - t_i)} + A_i$$

$$S_\Delta(t) = M_i \frac{(t_{i+1} - t)^3}{6(t_{i+1} - t_i)} + M_{i+1} \frac{(t - t_i)^3}{6(t_{i+1} - t_i)} + A_i(t - t_i) + B_i$$

mit Integrationskonstanten A_i, B_i.
Wir setzen $t_{i+1} - t_i =: h_{i+1}$. Aus (3.34) erhalten wir für A_i, B_i die folgenden Gleichungen

$$M_i \frac{h_{i+1}^2}{6} + B_i = y_i \Longrightarrow B_i = y_i - M_i \frac{h_{i+1}^2}{6}$$

$$M_{i+1} \frac{h_{i+1}^2}{6} + A_i h_{i+1} + B_i = y_{i+1} \Longrightarrow A_i = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{h_{i+1}}{6}(M_{i+1} - M_i)$$

Wir erhalten also für den Koeffizienten $(\alpha_i, \beta_i, \gamma_i, \delta_i)$ die Formeln:

$$\begin{array}{lll} \alpha_i & = S_{\Delta i}(t_i) & = y_i \\ \gamma_i & = S''_\Delta(t_i) & = \frac{M_i}{2} \\ \beta_i & = S'_\Delta(t_i) & = -\frac{M_i h_{i+1}}{2} + A_i \\ & & = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{2M_i + M_{i+1}}{6} h_{i+1} \\ \delta_i & = \frac{S'''_\Delta(t_{i+1})}{6} & = \frac{M_{i+1} - M_i}{6h_{i+1}} \end{array} \tag{3.38}$$

Damit haben wir S_Δ mit Hilfe der Momente ausgedrückt. Sobald neben den gegebenen Stützwerten y_k auch die zweiten Ableitungen M_k bekannt sind, sind die kubischen Polynome $S_{\Delta i}(t)$ in jedem Teilintervall eindeutig festgelegt und damit berechenbar. Mit diesen die Spline-Funktion beschreibenden Werten ist sowohl die Interpolationseigenschaft als auch die Stetigkeit der Funktion selbst und ihrer zweiten Ableitung in den inneren Stützpunkten sichergestellt. Es bleibt nur noch die Stetigkeit der ersten Ableitung an den inneren Stützstellen zu erfüllen. Für die erste Ableitung von $S_{\Delta i}(t)$ ergibt sich

$$S'_{\Delta i}(t) = \beta_i + 2\gamma_i(t - t_i) + 3\delta_i(t - t_i)^2 \tag{3.39}$$

Nach Substitution der Darstellungen (3.38) ergibt sich für die erste Ableitung am Ende des Intervalls

$$S'_{\Delta i}(t_{i+1}) = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{2M_{i+1} - M_i}{6} h_{i+1} \tag{3.40}$$

und nach Substitution des Indexwertes i durch $i - 1$

$$S'_{\Delta i-1}(t_i) = \frac{y_i - y_{i-1}}{h_i} - \frac{2M_i - M_{i-1}}{6} h_i. \tag{3.41}$$

Die Bedingung $S'_{\Delta,i-1}(t_i) = S'_{\Delta,i}(t_i)$ für eine innere Stützstelle t_i führt auf die Gleichung

$$\frac{h_i}{6} M_{i-1} + \frac{h_i + h_{i+1}}{3} M_i + \frac{h_{i+1}}{6} M_{i+1} = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \quad i = 1, \ldots, n-1 \tag{3.42}$$

wegen $S'_{\Delta,i}(t_i) = \beta_i$. Das sind $n - 1$ Bedingungen für die $n + 1$ Unbekannten $M_0, \ldots, M_n$. Aus den 3 möglichen Zusatzbedingungen erhalten wir:
a) aus (3.35)

$$M_0 = M_n = 0 \tag{3.43}$$

b) aus (3.36)

$$\begin{aligned} M_0 &= M_n && (3.44) \\ \frac{h_n}{6} M_{n-1} + \frac{h_n + h_1}{3} M_n + \frac{h_1}{6} M_1 &= \frac{y_1 - y_0}{h_1} - \frac{y_n - y_{n-1}}{h_n} && (3.45) \end{aligned}$$

c) aus (3.37)

$$\begin{aligned} \frac{h_1}{3} M_0 + \frac{h_1}{6} M_1 &= \frac{y_1 - y_0}{h_1} - y'_0 && (3.46) \\ \frac{h_n}{6} M_{n-1} + \frac{h_n}{3} M_n &= y'_n - \frac{y_n - y_{n-1}}{h_n} && (3.47) \end{aligned}$$

Dies ist ein Wust von Gleichungen, den man noch etwas vereinfachen kann, indem man ein paar Abkürzungen einführt.

$$\begin{aligned} \lambda_i &:= \frac{h_{i+1}}{h_i + h_{i+1}} \\ \mu_i &:= 1 - \lambda_i = \frac{h_i}{h_i + h_{i+1}} \qquad i = 1, \ldots, n-1 \\ d_i &:= \frac{6}{h_i + h_{i+1}} \left\{ \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \right\} \end{aligned}$$

Damit schreibt sich (3.42) als

$$\mu_i M_{i-1} + 2M_i + \lambda_i M_{i+1} = d_i \quad i = 1, \dots, n-1$$

Für die Zusatzbedingungen führen wir ein:

Im Fall a) $\lambda_0 := 0, d_0 := 0, \mu_n := 0, d_n := 0.$

Im Fall c) $\lambda_0 := 1, d_0 := \frac{6}{h_1}\left(\frac{y_1-y_0}{h_1} - y_0'\right), \mu_n := 1, d_n := \frac{6}{h_n}\left(y_n' - \frac{y_n-y_{n-1}}{h_n}\right).$
Dann erhalten wir in beiden Fällen das Gleichungssystem:

$$\begin{bmatrix} 2 & \lambda_0 & & & & & \\ \mu & 2 & \lambda_1 & & & & \\ & \mu_2 & 2 & \lambda_2 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \mu_{n-1} & 2 & \lambda_{n-1} \\ & & & & & \mu_n & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ \vdots \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ d_n \end{bmatrix} \tag{3.48}$$

Im Fall b) führen wir ein:

$$\begin{aligned} \lambda_n &:= \frac{h_1}{h_n + h_1} \\ \mu_n &:= 1 - \lambda_n = \frac{h_n}{h_n + h_1} \\ d_n &:= \frac{6}{h_n + h_1}\left\{\frac{y_1 - y_0}{h_1} - \frac{y_n - y_{n-1}}{h_n}\right\} \end{aligned}$$

und erhalten das Gleichungssystem

$$\begin{bmatrix} 2 & \lambda_1 & & & & & \mu_1 \\ \mu_2 & 2 & \lambda_2 & & & & \\ & \mu_3 & 2 & \lambda_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \mu_{n-1} & 2 & \lambda_{n-1} \\ \lambda_n & & & & & \mu_n & 2 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ \vdots \\ \vdots \\ M_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ \vdots \\ \vdots \\ d_n \end{bmatrix}. \tag{3.49}$$

Beide Gleichungssysteme (3.48), (3.49) sind eindeutig lösbar. Damit haben wir ein Verfahren die Splinefunktion zu bestimmen: Wir lösen (3.48) bzw. (3.49) und setzen die Werte in (3.38) ein, um die Koeffizienten für die Teilpolynome $S_{\Delta,i}$ zu erhalten. Splines haben sehr gute Konvergenzeigenschaften und eignen sich daher hervorragend zur Approximation von Funktionen bzw. zum Daten-Fitting.

Für manche Anwendungen ist es zweckmäßig, nicht wie hier einen Spline stückweise auf $[t_i, t_{i+1}]$ durch ein Polynom vom Grad k, sondern als Linearkombination

$$s(x) = \sum_{j=-k}^{n-1} c_j B_{kj}(x)$$

von $n + k$ linear unabhängigen Splines B_{kj} mit Koeffizienten c_j darzustellen. Einn solche Darstellung wird für sogenannte B-Splines besonders einfach. B-Splines sind bis auf Normierung durch die Bedingung

$$B_{kj}(x) = 0 \qquad x \notin [x_j, x_{j+k+1}]$$

eindeutig festgelegt.

Kapitel 4

Numerische Lösung von Anfangswertaufgaben

4.1 Allgemeine Problemstellung

Wir betrachten die gewöhnliche Differentialgleichung erster Ordnung (vgl. (1.4))

$$y'(t) = f(t, y(t)) \tag{4.1}$$

für t aus einem Intervall $[t_0, t_0 + a] \subseteq \mathbb{R}$. Dabei ist

$$\begin{array}{ll} y: & [t_0, t_0 + a] \longrightarrow \mathbb{R}^n \\ f: & [t_0, t_0 + a] \times G \longrightarrow \mathbb{R}^n \end{array}$$

wobei $G \subseteq \mathbb{R}^n$ eine geeignete Menge in $\mathbb{R}^n$ ist mit

$$y(t) \in G \quad \text{für} \quad t \in [t_0, t_0 + a].$$

Die Aufgabe, eine Funktion y zu bestimmen, für die

$$\begin{aligned} y'(t) &= f(t, y(t)) \text{ für } t \in [t_0, t_0 + a], \\ y(t_0) &= y_0 \end{aligned} \tag{4.2}$$

gilt für ein vorgegebenes $y_0 \in \mathbb{R}^n$, heißt Anfangswertaufgabe für die Differentialgleichung (4.1). y_0 heißt Anfangswert.
Eine gewöhnliche Differentialgleichung m-ter Ordnung

$$y^{(m)}(t) = g(t, y(t), y\prime(t), \ldots, y^{(m-1)}(t)),$$

($y : [t_0, t_0 + a] \longrightarrow \mathbb{R}^n$), kann in eine gewöhnliche Differentialgleichung erster Ordnung umgewandelt werden. Dazu setzt man

$$z_1(t) := y(t), \quad z_i(t) := z^{(i-1)}(t) \text{ für } i = 2, \ldots, m,$$

und

$$z(t) = \begin{pmatrix} z_1(t) \\ z_2(t) \\ \vdots \\ z_m(t) \end{pmatrix} \in \mathbb{R}^{mn}.$$

Damit gilt

$$z'(t) = \begin{pmatrix} z_1'(t) \\ z_2'(t) \\ \vdots \\ z_m'(t) \end{pmatrix} = \begin{pmatrix} z_2(t) \\ z_3(t) \\ \vdots \\ z_m(t) \\ g(t, z_1(t), \ldots, z_m(t)) \end{pmatrix} =: f(t, z(t)).$$

Wir haben hier wieder ein System erster Ordnung, allerdings für eine Funktion mit sehr vielen Komponentenfunktionen. Es genügt daher, gewöhnliche Differentialgleichungen der Form (4.1) zu betrachten.

Beispiel 4.1 *Gegeben sei die Differentialgleichung*

$$y'(t) = \lambda y(t) \qquad \text{für } t \in [0,1]$$

und $y(t) \in \mathbb{R}^n$. Die Lösung ist durch

$$y_c(t) = ce^{\lambda t} \qquad \text{mit } c \in \mathbb{R}^n$$

gegeben, dabei ist c beliebig. Fordern wir aber von y auch noch, daß

$$y(0) = y_0$$

für einen fest vorgegebenen Vektor $y_0 \in \mathbb{R}^n$, so muß $y(t)$ diejenige Funktion sein, für die

$$y(0) = ce^{\lambda 0} = c = y_0$$

gilt. Durch die Festlegung des Anfangswertes wird aus den unendlich vielen Lösungen der Differentialgleichungen genau eine ausgewählt.

In seltenen, einfachen Fällen wie etwa im Beispiel 4.1 kann man geschlossene analytische Lösungen einer Anfangswertaufgabe ermitteln. In den meisten Fällen, wie auch bei unserem Eingangsbeispiel 1 muß man numerisch eine gute Näherungslösung ermitteln. Ein numerisches Verfahren hat allerdings keine Chance, eine vernünftige Näherung zu berechnen, wenn die Lösung nicht eindeutig ist (erst recht natürlich nicht, wenn gar keine Lösung existiert). Darüberhinaus ist ein numerisches Verfahren auch hoffnungslos verloren, wenn die Lösung der Anfangswertaufgabe nicht stetig von den gegebenen Daten abhängt, d.h. wenn beliebig kleine Änderungen dazu führen, daß sich die Lösung völlig verändert.

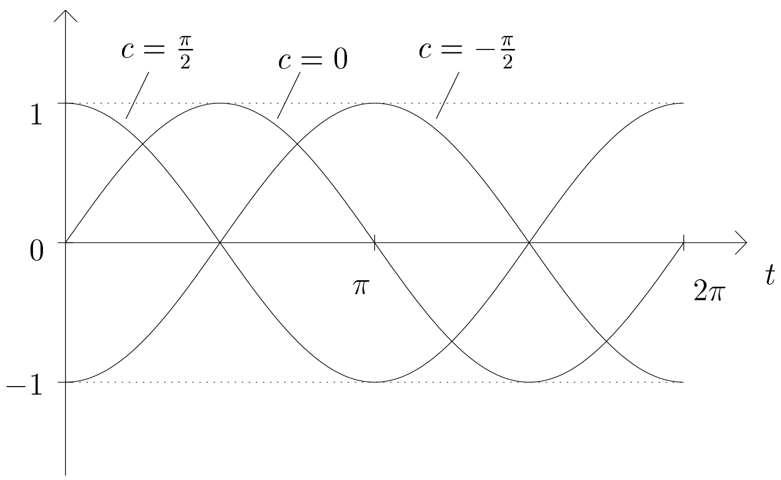


Abbildung 4.1: $\sin(t + c)$ für verschiedene Werte von c

Beispiel 4.2 *i)* $y' = \pm\sqrt{1 - y^2}, \quad t \in [t_0, \infty], t_o \geq 0,$
(positive oder negative Wurzel). Als Lösungen ergeben sich $y(t) = \sin(t + c)$ *für jedes* $c \in \mathbb{R}$*, sowie* $y \equiv 1$ *und* $y \equiv -1$ *und Kombinationen aus diesen Funktionen (siehe Abbildung 4.1). Je nach Wahl der Anfangsbedingung ist diese Anfangswertaufgabe eindeutig, nicht eindeutig oder nicht lösbar:*

Für $y(t_0) = 4$ *gibt es keine Lösung,* t_0 *beliebig.*
Für $y(0) = 1$ *lauten die Lösungen:* $y(t) = \sin(t + \frac{\pi}{2})$ *und* $y = 1$.
Für $y(0) = 0$ *lautet die Lösung:* $y(t) = \sin(t)$.

ii) $y' = 10(y - \frac{t^2}{1+t^2}) + \frac{2t}{(1+t^2)^2} \qquad y(0) = y_0.$

Als Lösung ergibt sich: $y(t) = y_0 e^{10t} + \frac{t^2}{1+t^2}.$

Für $y_0 = 0$ *lautet die Lösung:* $\frac{t^2}{1+t^2} = \hat{y}(t)$

Für $y_0 = -\varepsilon$ *lautet die Lösung:* $-\varepsilon e^{10t} + \frac{t^2}{1+t^2} = \tilde{y}(t)$

Hier existiert zu jedem ε *eine eindeutige Lösung und es gilt für einen Zeitpunkt* t*:*

$$|\tilde{y}(t) - \hat{y}(t)| = |e^{10t}|\ |\varepsilon|.$$

Die Lösung ändert sich also in stetiger Abhängigkeit von ε*, aber der Faktor* $|e^{10t}|$ *wächst sehr schnell mit* t*. Nur lokal (d.h. in der Nähe des Anfangspunktes* $t = 0$*) ist es richtig, daß bei kleiner Abweichung* ε *auch die Lösung nur sehr wenig abweicht*

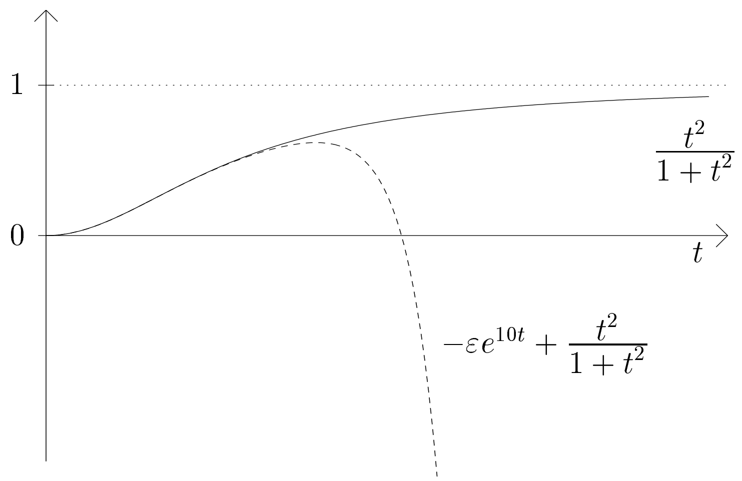


Abbildung 4.2: $y(t) = y_0 e^{10t} + \frac{t^2}{1+t^2}$

(siehe Abbildung 4.2). Der "Fehler" wächst exponentiell an. Dies ist eine typische Erscheinung bei Anfangswertaufgaben.

Existenz und Eindeutigkeit sowie die stetige Abhängigkeit von den Daten sind notwendige Bedingungen, die eine Anfangswertaufgabe für eine gewöhnliche Differentialgleichung erfüllen muß, damit sie einer vernünftigen numerischen Behandlung zugänglich ist. Es gibt eine Reihe mathematischer Sätze aus der Theorie gewöhnlicher Differentialgleichungen, die Voraussetzungen angeben, unter denen diese Bedingungen erfüllt sind. Wir wollen hier nur zwei spezielle Sätze dieser Art ohne Beweis angeben, da sie die sogenannte Lipschitzbedingung einführen, der im Zusammenhang mit der numerischen Behandlung gewöhnlicher Differentialgleichungen eine gewisse Bedeutung zukommt.

Definition 4.3 *Sei $f : [t_0, t_0 + a] \times G \longrightarrow \mathbb{R}^n$ stetig. Sei $R = \{(t, y) | t_0 \le t \le t_0 + a, \|y - y_0\| \le b\}$, wobei $b > 0$. Solch ein R bezeichnet man häufig als Streifen, insbesondere wenn $b = -\infty$. f genügt einer Lipschitzbedingung auf R (auch: f ist Lipschitz-stetig auf R), genau dann wenn es eine Konstante L gibt, so daß für alle $(t, y_1), (t, y_2) \in R$ gilt:*

$$\|f(t, y_1) - f(t, y_2)\| \le L \|y_1 - y_2\|. \tag{4.3}$$

Bemerkung 4.4 *Falls alle $\frac{\partial f_i}{\partial y_j}$ existieren und auf R stetig und beschränkt sind, so ist die*

Lipschitzbedingung (4.3) erfüllt. Denn nach dem Mittelwertsatz gilt:

$$\begin{aligned}\|f(t,y_1)-f(t,y_2)\| &\leq \|\int_0^1 D_y f(t,y_1+\xi(y_1-y_2))(y_1-y_2)d\xi\| \\ &\leq \int_0^1 \|D_y f(t,y_1+\xi(y_1-y_2))\|\|y_1-y_2\|d\xi \leq L\|y_1-y_2\|,\end{aligned}$$

wobei L das Maximum von $\|D_y f(t,\eta)\|$ auf R ist. Ganz kurz gesagt: Hinreichende glatte Funktionen f sind Lipschitz-stetig.

Der folgende Satz, der hier ohne Beweis angegeben wird, sagt, daß die Lipschitzbedingung im wesentlichen die Existenz und Eindeutigkeit der Lösung der Anfangswertaufgabe sichert.

Satz 4.5 *(Satz von Picard und Lindelöf)*
Sei $f:[t_0,t_0+a]\times G\longrightarrow\mathbb{R}^n$ stetig auf $R=\{(t,y)|t_0\leq t\leq t_0+a,\|y-y_0\|\leq b\}$, und genüge f auf R einer Lipschitzbedingung. Sei

$$M:=\max_{(t,y)\in R}\|f(t,y)\|,\quad \alpha:=\min\{a,\frac{b}{M}\}.$$

Dann hat die Anfangswertaufgabe

$$y'=f(t,y),\quad y(t_0)=y_0$$

eine eindeutige Lösung $y(t)$ auf $[t_0,t_0+\alpha]$.

Bemerkung 4.6 *Man beachte, daß die Existenz und Eindeutigkeit der Lösung nur lokal gesichert ist, d.h. hier nur für $t\leq t_0+\alpha\leq t_0+a$. Da $\|y(t)-y_0\|=\|\int_{t_0}^t f(s,y(s))ds\|\leq M(t-t_0)$, ist $(t,y(t))$ für $t-t_0>\frac{b}{M}$ eventuell nicht mehr in R.*

Eine Aussage über die Abhängigkeit der Lösung von Abweichungen im Anfangswert gibt der folgende Satz, der hier ebenfalls ohne Beweis zitiert wird. Seien die Voraussetzungen des Satzes von Picard und Lindelöf erfüllt, wobei allerdings R durch $R_D=\{(t,y)|t_0\leq t\leq t_0+a,\|y-z\|\leq b, z\in D\}$ ersetzt sei und D eine Umgebung des Anfangswertes y_0 ist. Dann existiert nach dem vorangegangenen Satz für jedes $z\in D$ eine eindeutige Lösung $y_z(t)$ der Anfangswertaufgabe

$$y'(t)=f(t,y(t)),\quad y(t_0)=z.$$

Satz 4.7 *Mit obigen Voraussetzungen gilt:*

$$\|y_{z_1}(t)-y_{z_2}(t)\|\leq e^{L(t-t_0)}\|z_1-z_2\|$$

für alle $t\in[t_0,t_0+\alpha]$. Dabei ist L die Lipschitzkonstante von f auf R_D.

4.2 Allgemeine Einschrittverfahren

Wir wollen die Anfangswertaufgabe

$$\begin{aligned} y' &= f(t, y(t)), \quad \text{für} \quad t_0 \in [t_0, t_0 + a] && (4.4) \\ y(t_0) &= y_0 && (4.5) \end{aligned}$$

numerisch lösen. Eine naheliegende, einfache Idee ist die folgende.

- Unterteile das Intervall $[t_0, t_0 + a]$ mittels äquidistanter Punkte

  $$t_0 < t_1 < t_2 < \cdots < t_N = t_0 + a,$$

  wobei $t_i = t_0 + ih$ für $i = 0, \ldots, N$ und $h = \frac{a}{N}$. Die t_i nennt man Knoten oder Gitterpunkte, h heißt Schrittweite.

- Betrachte die Differentialgleichung nur in den Knoten, also

  $$y'(t_i) = f(t_i, y(t_i)) \quad \text{für } i = 0, \ldots, N \tag{4.6}$$

  und nähere die Ableitung durch eine geeignete Differenz der $y(t_i)$. Aus der Taylorentwicklung von $y(t_{i+1}) = y(t_i + h)$ um t_i,

  $$y(t_{i+1}) = y(t_i + h) = y(t_i) + hy'(t_i) + \frac{h^2}{2} y''(t_i) + \text{ Terme höherer Ordnung,}$$

  erhält man z.B.:

  $$y'(t_i) = \frac{y(t_{i+1}) - y(t_i)}{h} + \mathcal{O}(h), \tag{4.7}$$

  wobei $\mathcal{O}(h)$ die Zusammenfassung aller restlichen Terme $(\frac{h^2}{2} y''(t_i) + \cdots)$ bezeichnet. Setzt man nun (4.7) in (4.6) ein, so ergibt sich

  $$\frac{y(t_{i+1}) - y(t_i)}{h} = f(t_i, y(t_i)) + \mathcal{O}(h)$$

  bzw. umgeformt

  $$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \mathcal{O}(h^2).$$

- Läßt man nun die Terme $\mathcal{O}(h^2)$ weg, so erfüllen die $y(t_i)$ diese Gleichung nicht mehr exakt. Aber da $\mathcal{O}(h)$ klein ist für kleines h, können wir hoffen, daß die Größen, die diese Gleichung nun erfüllen, gute Näherungen an die $y(t_i)$ sind.

  Man betrachtet also $u_0, \ldots, u_N \in \mathbb{R}^n$, für die gilt:

  $$\begin{aligned} u_0 &= y_0 \\ u_{i+1} &= u_i + hf(t_i, u_i) \text{ für } i = 0, \ldots, N-1 && (4.8) \end{aligned}$$

Diese Werte kann man, ausgehend von $u_0 = y_0$ offensichtlich sehr einfach berechnen. Für alle $i = 0, \ldots, N$ sehen wir u_i als Approximation an $y(t_i)$ an. Wir werden später genauer sehen, daß die Näherungswerte u_i gegen die entsprechenden Werte der Lösung y konvergieren, wenn die Schrittweite h gegen Null geht. (4.8) beschreibt das explizite Eulerverfahren.

<u>Explizites Eulerverfahren</u>

$$\begin{aligned} \text{Setze} \quad h &:= \frac{a}{N}, \quad t_i := t_0 + ih \\ u_0 &:= y_0 \\ u_{i+1} &:= u_i + hf(t_i, u_i) \quad \text{für} \quad i = 0, \ldots, N-1 \end{aligned}$$

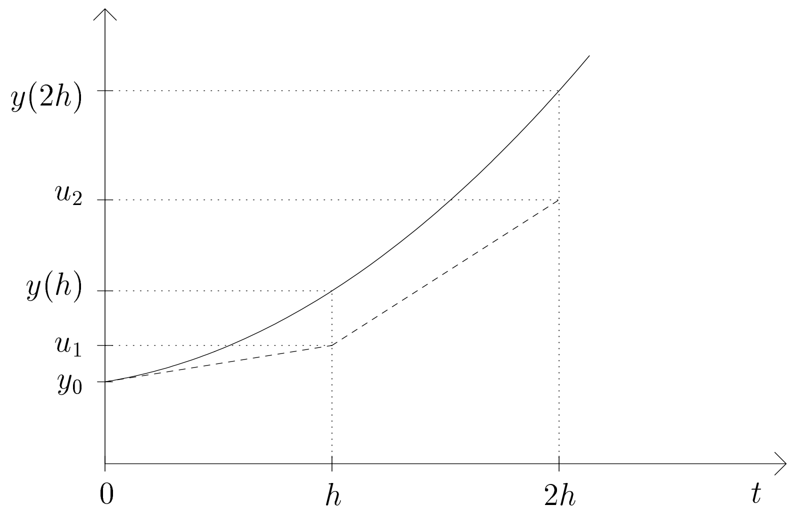
Das Eulerverfahren ist der einfachste und älteste Vertreter der sogenannten expliziten Einschrittverfahren. Dabei ersetzt man $f(t_i, u_i)$ durch eine allgemeinere Funktion $\phi(t_i, u_i, h, f)$, die Inkrementfunktion. Man erhält:

<u>Allgemeines explizites Einschrittverfahren:</u>

$$\begin{aligned} \text{Setze} \quad h &:= \frac{a}{N}, \quad t_i := t_0 + ih \\ u_0 &:= y_0 \\ u_{i+1} &:= u_i + h\phi(t_i, u_i, h, f) \quad \text{für} \quad i = 0, \ldots, N-1 \end{aligned}$$

Man wird dabei ϕ so wählen, daß die u_i möglichst nahe bei $y(t_i)$ liegen. Wir werden dies gleich genauer betrachten.

Bemerkung 4.8 *Geometrisch läßt sich das Eulerverfahren folgendermaßen deuten:*



Denn:

$$u_1 = y_0 + hf(t_0, y_0) = y_0 + hy'(t_0)$$

d.h. das Inkrement, um das y_0 vergrößert wird, um u_1 zu ergeben ist gerade $h\cdot$ $\{$Steigung der Tangente von y in $t_0\}$. Weiter ist $u_2 = u_1 + hf(t_1, u_1)$, also das Inkrement $h \cdot$ $\{$Näherung an die Tangentensteigung von y in $t_1\}$, usw.

4.2.1 Fehlerbetrachtungen und Konvergenz

Wann konvergiert ein solches Einschrittverfahren? Wie gut approximieren die u_i die exakten Werte $y(t_i)$? Zur Beantwortung dieser Fragen benötigen wir eine Abschätzung für den globalen Diskretisierungsfehler:

$$e_j = y(t_j) - u_j \text{ für } j = 0, 1, 2, \cdots$$

Um e_j abschätzen zu können, muß man zunächst den lokalen Fehler betrachten. Das ist der Fehler an einer Stelle $\hat{t} \in [t_0, t_0 + a]$ in nur einem Schritt bei exaktem Anfangswert $y(\hat{t})$. Sei also $y(\hat{t}) = z$ und stellen wir uns vor, wir setzen das Verfahren in dem Punkt $\hat{t}$ als Anfangspunkt an. Dann ist die exakte Steigung, die von $y(\hat{t})$ nach $y(\hat{t} + h)$ führt:

$$\triangle(\hat{t}, z, h, f) := \begin{cases} \dfrac{y(\hat{t}+h) - y(\hat{t})}{h} & \text{für} \quad h \neq 0 \\ f(\hat{t}, z) & \text{für} \quad h = 0 \end{cases} .$$

Dies ist der Differenzenquotient von y zur Schrittweite h bei $\hat{t}$.
Für die Gitterfunktion

$$u_h : \{t_0, t_1, \ldots, t_N\} \longrightarrow \mathbb{R}^n, \quad u_h(t_j) = u_j, \quad J = 0, 1, \ldots, N$$

haben wir entsprechend $u_h(\hat{t}) = z$ und als Näherung obiger Steigung verwendet man:

$$\frac{u_h(\hat{t}+h) - u_h(\hat{t})}{h} = \phi(\hat{t}, z, h, f),$$

wie man aus dem Algorithmus leicht ersieht. Die Größe

$$\tau(\hat{t}, z, h, f) = \triangle(\hat{t}, z, h, f) - \phi(\hat{t}, z, h, f)$$

heißt lokaler Diskretisierungsfehler. Wegen

$$\underbrace{y(\hat{t}+h) = u_h(\hat{t}) + h\phi(\hat{t}, u_h(\hat{t}), h, f)}_{\text{Verfahrensgleichung für } y(\hat{t})} \quad + \qquad \underbrace{h\tau(\hat{t}, z, h, f)}_{\text{Fehler } y(\hat{t}+h) - u_h(\hat{t}+h)}$$

gibt er an, wie gut die exakte Lösung die Gleichung des Einschrittverfahrens erfüllt. τ ist ein Maß für die lokale Güte des Verfahrens.

Damit das Einschrittverfahren vernünftig ist, muß sicherlich gelten

$$\lim_{h\to 0} \tau(\hat{t}, z, h, f) = 0. \tag{4.9}$$

Da aber $\lim_{h\to 0} \triangle(\hat{t}, z, h, f) = y'(\hat{t}) = f(\hat{t}, z)$ ist, ist dies äquivalent zu

$$\lim_{h\to 0} \phi(\hat{t}, z, h, f) = f(\hat{t}, z).$$

Man nennt ein Einschrittverfahren konsistent, wenn für alle $\hat{t} \in [t_0, t_0 + a]$, und alle f mit stetigen und beschränkten partiellen Ableitungen, (4.9) gilt.

Beispiel 4.9 *Das Eulerverfahren ist konsistent, falls f genügend glatt ist. Um dies zu sehen, betrachtet man die Taylorentwicklung*

$$y(\hat{t} + h) = y(\hat{t}) + hy'(\hat{t}) + \frac{h^2}{2} y''(\hat{t}) + \mathcal{O}(h^3).$$

Beachte daß $y(\hat{t}) = z,\ y'(\hat{t}) = f(\hat{t}, y(\hat{t}))$, also

$$\begin{aligned} y''(\hat{t}) &= \frac{d}{dt} f(t, y(t))|_{t=\hat{t}} \\ &= f_t(t, y(t))|_{t=\hat{t}} + f_y(t, y(t)) \cdot y'(t)|_{t=\hat{t}} \\ &= f_t(\hat{t}, z) + f_y(\hat{t}, z) f(\hat{t}, z). \end{aligned}$$

Dabei bezeichnet f_t die partielle Ableitung von f nach t und f_y entsprechend die partielle Ableitung nach y. Also erhält man

$$\begin{aligned} \triangle(\hat{t}, z, h, f) &= \frac{y(\hat{t} + h) - y(\hat{t})}{h} \\ &= y'(\hat{t}) + \frac{h}{2} y''(\hat{t}) + \mathcal{O}(h^2) \\ &= f(\hat{t}, z) + \frac{h}{2}[f_t(\hat{t}, z) + f_y(\hat{t}, z) f(\hat{t}, z)] + \mathcal{O}(h^2). \end{aligned}$$

Beim Eulerverfahren ist $\phi(\hat{t}, z, h, f) = f(\hat{t}, z)$. Also folgt für den lokalen Diskretisierungsfehler:

$$\begin{aligned} \tau(\hat{t}, z, h, f) &= \triangle(\hat{t}, z, h, f) - \phi(\hat{t}, z, h, f) \\ &= \frac{h}{2}[f_t(\hat{t}, z) + f_y(\hat{t}, z) f(\hat{t}, z)] + \mathcal{O}(h^2) \\ &= \mathcal{O}(h) \longrightarrow 0 \text{ für } h \longrightarrow 0. \end{aligned}$$

Man sagt, ein Einschrittverfahren ist konsistent von der Ordnung p, wenn $\tau(\hat{t}, z, h, f) = \mathcal{O}(h^p)$ für alle $\hat{t} \in [t_0, t_0 + a]$, alle $z \in \mathbb{R}$ und alle genügend glatten f. Das Eulerverfahren ist also konsistent von der Ordnung 1.

Verfahren höherer Ordnung kann man gewinnen, indem man $\phi(\hat{t}, z, h, f)$ so bestimmt, daß die Taylorentwicklung von $\tau(\hat{t}, z, h, f)$ mit einer möglichst hohen Potenz von h beginnt. Ist etwa $\tau = \mathcal{O}(h^2)$ so heißt das ja z.B., daß bei Verkleinerung von h auf $\frac{1}{10}h$, sich der lokale Diskretisierungsfehler τ auf ein Hundertstel reduziert, d.h. grob gesprochen man gewinnt zwei korrekte Stellen mehr, und dies ist sicher vorteilhaft.
Einfache Verfahren höherer Ordnung erhält man mit dem Ansatz

$$\phi(t, z, h, f) = a_1 f(t, z) + a_2 f(t + p_1 h, z + p_2 h f(t, z))$$

Man bestimmt nun a_1, a_2, p_1, p_2 so, daß die Entwicklung von τ mit möglichst hoher $h-$Potenz beginnt. Die Taylorentwicklung von $\phi(t, z, h, f)$ ergibt

$$\phi(t, z, h, f) = (a_1 + a_2) f(t, z) + a_2 h [p_1 f_t(t, z) + p_2 f_z(t, z) f(t, z)] + O(h^2).$$

Für Verfahren 2-ter Ordnung ergibt daher sich:

$$a_1 + a_2 = 1 \quad a_2 p_1 = \frac{1}{2} \quad a_2 p_2 = \frac{1}{2}.$$

Lösungen sind:

1. $a_1 = \frac{1}{2}, a_2 = \frac{1}{2}, p_1 = p_2 = 1$
   <u>Verfahren von Heun (1900)</u>

$$\begin{aligned} u_0 &= y_0 \\ u_{i+1} &= u_i + \frac{h}{2}[f(t_i, u_i) + f(t_{i+1}, u_i + h f(t_i, u_i))] \end{aligned}$$

2. $a_1 = 0, a_2 = 1, p_1 = p_2 = \frac{1}{2}$
   <u>Modifiziertes Eulerverfahren von Collatz (1960)</u>

$$\begin{aligned} u_0 &= y_0 \\ u_{i+1} &= u_i + h f(t_i + \frac{h}{2}, u_i + \frac{h}{2} f(t_i, u_i)) \end{aligned}$$

Bemerkung 4.10 *Bei den beiden obigen Verfahren benötigt man pro Schritt 2 Auswertungen der Funktion f. Diese Auswertungen sind sicher die "teuersten" Berechnungen im Schritt. Je nach Problemstellung muß jede Auswertung unter Umständen durch Messungen erfolgen und ist damit außerordentlich aufwendig. Man möchte daher die Anzahl der Funktionsauswertungen klein halten.*

Bis jetzt haben wir gesehen, daß bei einem konsistenten Einschrittverfahren der lokale Fehler gegen Null geht, wenn die Schrittweite h gegen Null geht. Dabei hat man aber nur einen Schritt im Zeitintervall mit exakten Anfangsdaten betrachtet. Dieser Fehler, den man schon beim Schritt von t_0 nach t_1 macht, setzt sich im nächsten, von t_1 nach t_2, dadurch fort, daß man jetzt mit dem Näherungswert $u_h(t_1) = u_1$ arbeiten muß und nicht mehr den exakten Wert $y(t_1)$ zur Verfügung hat usw.. Es muß geklärt werden, ob diese sich in alle weiteren Schritte fortpflanzenden Fehler nicht verhindern, daß der globale Fehler klein wird. Der folgende Satz gibt darüber Auskunft.

Satz 4.11 *ϕ genüge auf der Menge $[t_0, t_0 + a] \times \mathbb{R}^n$ einer Lipschitzbedingung bzgl. y mit Lipschitzkonstante L. Dann gilt für den globalen Fehler*

$$e_j = y(t_j) - u_j \qquad j = 0, \ldots, N$$

folgende Abschätzung:

$$|e_n| \leq (|e_0| + (t_n - t_0)\tau_h)e^{L(t_n - t_0)} \text{ für } n = 0, \ldots, N$$

Dabei ist $\tau_h = max_j|\tau_j|$ und τ_j ist der lokale Diskretisierungsfehler bei t_j.

Die Lipschitzbedingung für ϕ ergibt sich im allgemeinen aus der Lipschitzbedingung für f. Im Falle des Eulerverfahrens ist es sogar genau die Lipschitzbedingung für f. e_0 bezeichnet hier einen möglichen Fehler in den Anfangsdaten. In Beispiel 4.2 ii) hatten wir schon gesehen, daß ein solcher Fehler mit einem exponentiell wachsenden Faktor ins Ergebnis eingehen kann.
Das obige Ergebnis besagt grob gesprochen: Der globale Fehler verhält sich bei expliziten Einschrittverfahren wie der lokale. Ist also $\tau_h = \mathcal{O}(h^p)$, so ist auch $e_n = \mathcal{O}(h^p)$. Man beachte aber, daß die Konstante $e^{L(t_n - t_0)}$ mit wachsenden t_n sehr groß werden kann.
Betrachtet man ein beliebiges festes $\hat{t} \in [t_0, t_0 + a]$ und sind die Schrittweiten $h_1, h_2, \ldots$ so gewählt, daß $\hat{t}$ stets ein Knoten t_{n_1}, t_{n_2} usw. ist, so ist $n_k = \frac{\hat{t} - t_0}{h_k}$. Für $h_k \longrightarrow 0$ (für $k \to \infty$) geht also $n_k \to \infty$ und bei einem konvergenten Einschrittverfahren daher die Näherungswerte $u_{n_1}, u_{n_2}, \ldots$ gegen $y(\hat{t})$. D.h. gerade $\lim_{n\to\infty} |e_{n_k}| = 0$. Man nennt ein solches Einschrittverfahren konvergent, d.h. wenn

$$\lim_{n\to\infty} |e_n| = 0.$$

Wie wir uns gerade überlegt haben, sind konsistente Einschrittverfahren sind konvergent. Die Konvergenz ist dabei von derselben Ordnung wie der lokale Diskretisierungsfehler.
In der Praxis stellt sich natürlich die Frage:" Wie klein muß h sein, damit wir eine bestimmte von uns gewünschte Genauigkeit erhalten?" Da die Größen L und τ_h selbst beim Eulerverfahren schwer zugänglich sind, gibt der obige Satz hierauf leider keine Antwort.
Das folgende Beispiel veranschaulicht, wie sich $\tau_h = \mathcal{O}(h_p)$ im Ergebnis bei Schrittweitenverkleinerung auswirkt.

Beispiel 4.12 *Wir wollen uns die drei Verfahren Euler, Heun und modifiziertes Eulerverfahren im praktischen Vergleich anschauen. Dazu betrachten wir die Anfangswertaufgabe*

$$y' = -2ty^2, \quad y(0) = 1$$

mit der exakten Lösung $y(t) = 1/(t^2 + 1)$. Mit der Methode von Euler erhalten wir die in der Tabelle 4.1 zusammengestellten Näherungswerte u_k für verschiedene Schrittweiten h an gleichen Stellen t_k sowie die zugehörigen Fehler $e_k := y(t_k) - u_k$. Der Fehler nimmt etwa proportional zur Schrittweite h ab.

		$h = 0.1$		$h = 0.01$		$h = 0.001$	
t_k	$y(t_k)$	u_k	$\lvert e_k\rvert$	u_k	$\lvert e_k\rvert$	u_k	$\lvert e_k\rvert$
0	1.00000	1.00000		1.00000		1.00000	
0.1	0.99010	1.00000	0.00990	0.99107	0.00097	0.99020	0.00010
0.2	0.96154	0.98000	0.01846	0.96330	0.00176	0.96171	0.00018
0.3	0.91743	0.94158	0.02415	0.91969	0.00226	0.91766	0.00022
0.4	0.86207	0.88839	0.02632	0.86448	0.00242	0.86231	0.00024
0.5	0.80000	0.82525	0.02525	0.80229	0.00229	0.80023	0.00023
0.6	0.73529	0.75715	0.02185	0.73727	0.00198	0.73549	0.00020

Tabelle 4.1: Verfahren von Euler

Obige Anfangswertaufgabe ist mit dem Verfahren von Heun und dem modifizierten Eulerverfahren je mit den Schrittweiten $h = 0.1$ und $h = 0.05$ behandelt worden. Die erhaltenen Näherungswerte u_k sind zusammen mit den zugehörigen globalen Fehlern $e_k = y(t_k) - u_k$ in der Tabelle 4.2 zusammengestellt. Die Resultate offenbaren die bedeutend kleineren Diskretisierungsfehler der beiden Methoden und ihre Fehlerordnung 2, d.h. die quadratische Fehlerverkleinerung.
Die Abbildung 4.3 zeigt die tabillierten Daten nochmals grafisch.

Schließlich müssen wir noch berücksichtigen, daß wir auf einem Rechner bei der Ermittlung der u_k Rundungsfehler machen, während die obigen Überlegungen von exakter Rechnung ausgehen. Der hier abschließend angegebene Satz besagt, daß es nicht sinnvoll ist, bei Durchführung des Verfahrens auf einem Rechner, die Schrittweite h über eine gewisse Grenze hinaus zu verkleinern, da sonst die Rundungsfehlereinflüsse das Ergebnis stark verfälschen können.

Satz 4.13 *(Rundungsfehlereinfluß beim expliziten Einschrittverfahren)*
Seien u_i die (theoretischen) Verfahrenswerte, die gemäß

$$u_{i+1} = u_i + h\phi(t_i, u_i, h, f), \quad u_0 = y_0$$

gewonnen werden, und seien $\tilde{u}_i$ die vom Rechner hierfür gelieferten Werte, für die gilt

$$\tilde{u}_{i+1} = \tilde{u}_i + h\phi(t_i, \tilde{u}_i, h, f) + \varepsilon_{i+1}, \ \tilde{u}_0 = u_0 + \varepsilon_0.$$

ϕ genüge einer Lipschitzbedingung mit Konstante L bezgl. u und es sei $|\varepsilon_i| \leq \varepsilon$ für alle $i \geq 0$. Dann gilt für die "Rundungsfehler" $r_i = \tilde{u}_i - u_i$:

$$|r_n| \leq e^{L(t_n - t_0)}(|r_0| + \frac{t_n + t_0}{h}\varepsilon).$$

Bemerkung 4.14 *Wir müssen also einerseits h klein wählen, damit der globale Fehler klein wird. Andererseits wird, wenn h zu winzig ist, der Ausdruck $\frac{t_n - t_0}{h}\varepsilon$ zu groß. Der von den Rundungsfehlern herrührende Fehleranteil kann dann das Ergebnis stark verfälschen. Dies ist ein Grund, weshalb man Verfahren höherer Ordnung benötigt.*

Modifiziertes Eulerverfahren Methode von Heun

	$h = 0.1$		$h = 0.05$		$h = 0.1$		$h = 0.05$	
t_k	u_k	$\|e_k\|$	u_k	$\|e_k\|$	u_k	$\|e_k\|$	u_k	$\|e_k\|$
0	1.00000		1.00000		1.00000		1.00000	
0.1	0.99000	0.00010	0.99007	0.00002	0.99000	0.00010	0.99009	0.00001
0.2	0.96118	0.00036	0.96145	0.00009	0.96137	0.00017	0.96152	0.00002
0.3	0.91674	0.00069	0.91727	0.00016	0.91725	0.00019	0.91742	0.00001
0.4	0.86110	0.00096	0.86184	0.00023	0.86195	0.00011	0.86208	0.00001
0.5	0.79889	0.00111	0.79974	0.00026	0.80003	0.00003	0.80004	0.00004
0.6	0.73418	0.00111	0.73503	0.00026	0.73553	0.00023	0.73538	0.00009
0.7	0.67014	0.00100	0.67091	0.00023	0.67159	0.00045	0.67128	0.00014
0.8	0.60895	0.00080	0.60957	0.00018	0.61040	0.00064	0.60993	0.00018
0.9	0.55191	0.00058	0.55236	0.00013	0.55329	0.00080	0.55270	0.00021
1.0	0.49964	0.00036	0.49992	0.00008	0.50092	0.00092	0.50024	0.00024

Tabelle 4.2: modifiziertes Eulerverfahren und Methode von Heun

4.3 Explizites und implizites Eulerverfahren

Bisher haben wir

$$y(t+h) = y(t) + h\underbrace{(f(t,y) + \ldots)}_{\phi(t,y(t),h,f)}$$

betrachtet und damit das explizite Einzelschrittverfahren

$$u_{i+1} = u_i + h\phi(t_i, u_i, h, f)$$

entwickelt.
Wählt man bei der obigen Entwicklung $h < 0$ (Rückwärtsschritt), dann müßte die entsprechende Verfahrensgleichung lauten:

$$u_{i-1} = u_i + h\phi(t_i, u_i, h, f)$$

bzw.

$$u_{i+1} = u_i - h\phi(t_{i+1}, u_{i+1}, h, f).$$

Dies nennt man implizites Einzelschrittverfahren. Ein Beispiel hierfür ist:

$$u_{i+1} = u_i + hf(t_{i+1}, u_{i+1})$$

mit $h > 0$ das implizite Eulerverfahren (oder auch backward Eulerverfahren). Im expliziten Fall erhalten wir u_{i+1} direkt, im impliziten Fall müssen wir ein im allgemeinen nichtlineares Gleichungssystem lösen.

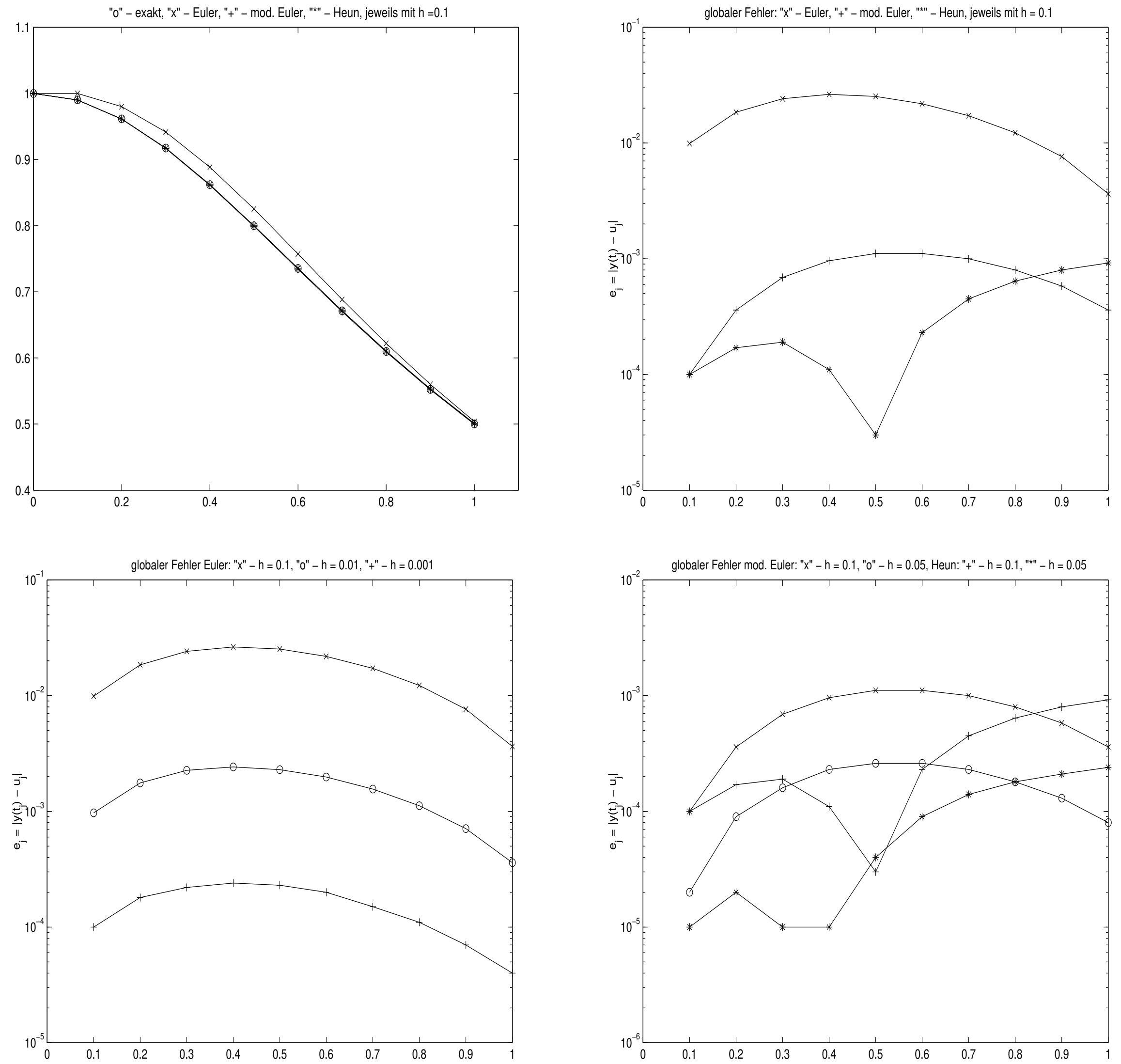


Abbildung 4.3:

Beispiel 4.15 *Betrachte die Anfangswertaufgabe*

$$y' = \lambda y(t), \quad y(0) = 1, \tag{4.10}$$

mit der Lösung $y(t) = e^{\lambda t}$ *für* $\lambda = -25$. *Die Abbildung 4.4 zeigt die mit dem expliziten und dem impliziten Eulerverfahren berechneten Lösungen bei der Schrittweitenwahl* $h = 0.1$ *und* $h = 0.01$. *Bei der Wahl* $h = 0.01$ *scheinen beide Verfahren die gesuchte Lösung gut anzunähren, aber bei der Wahl* $h = 0.1$ *oszilliert die mittels des expliziten Eulerverfahrens berechnete Lösung stark, während die exakte Lösung fast* 0 *ist.*

Wodurch ist das unterschiedliche Verhalten bedingt? Dazu betrachtet man die jeweiligen

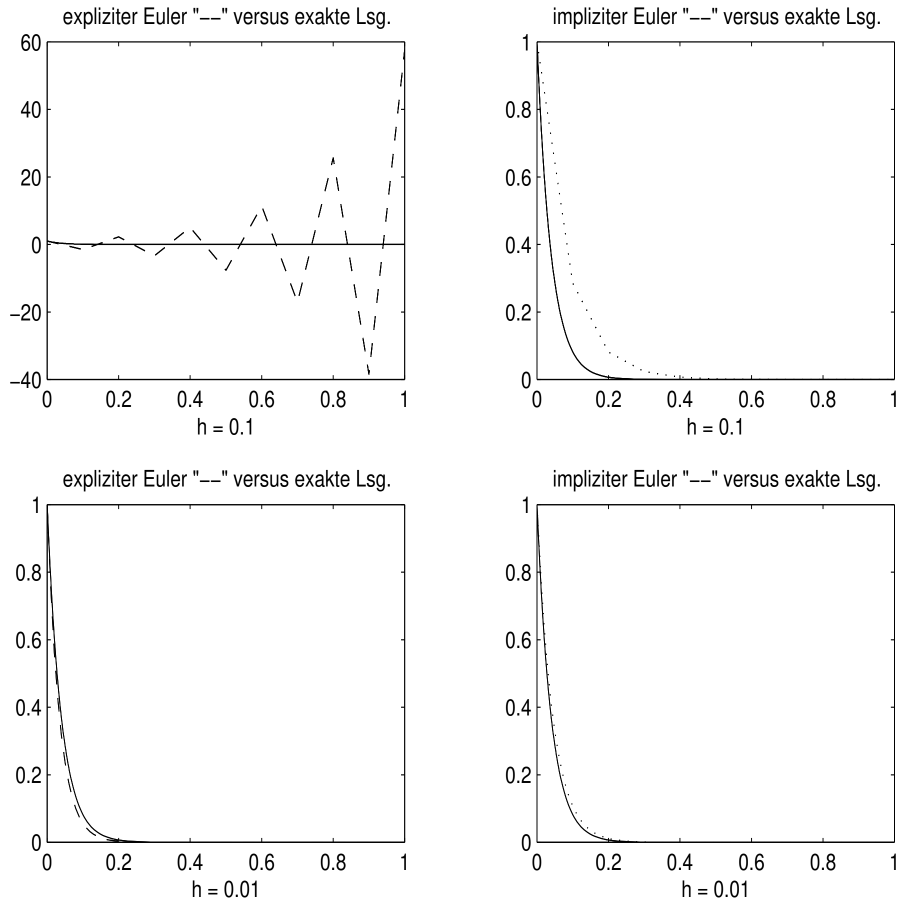


Abbildung 4.4: explizites und implizites Eulerverfahren

Verfahrensvorschriften genauer: Beim expliziten Eulerverfahren haben wir

$$u_{i+1} = u_i + hf(t_i, u_i),$$

was für die Anfangswertaufgabe (4.10) zu

$$u_{i+1} = (1 + \lambda h)u_i$$

wird. Beim impliziten Eulerverfahren haben wir

$$u_{i+1} = u_i + hf(t_{i+1}, u_{i+1}),$$

woraus wir für die Anfangswertaufgabe (4.10)

$$u_{i+1} = \frac{1}{1 - \lambda h} u_i$$

erhalten. In beiden Fällen kann die Verfahrensgleichung also in der Form

$$u_{i+1} = F(\lambda h)u_i$$

geschrieben werden, mit

$$F_E(\lambda h) = 1 + \lambda h$$

beim expliziten Eulerverfahren und

$$F_I(\lambda h) = \frac{1}{1 - \lambda h}$$

beim impliziten Eulerverfahren. Für die Lösung der Anfangswertaufgabe (4.10) gilt

$$y(t_{i+1}) = y(t_i + h) = e^{\lambda(t_i+h)} = e^{\lambda h} e^{\lambda t_i} = e^{\lambda h} y(t_i).$$

Betrachtet man die Taylorreihe für $e^{\lambda h}$

$$e^{\lambda h} = 1 + h\lambda + \frac{1}{2}h^2\lambda^2 + \frac{1}{6}h^3\lambda^3 + \frac{1}{24}h^4\lambda^4 + \dots\ ,$$

so sieht man, daß $F_E(\lambda h)$ gerade mit den ersten beiden Termen dieser Taylorreihe übereinstimmt. Der Multiplikator $F_E(\lambda h)$ stellt für betragskleine $h\lambda$ sicher eine gute Approximation an $e^{\lambda h}$ dar und man erwartet, daß das explizite Eulerverfahren dann die Anfangswertaufgabe (4.10) gut löst.

Betrachten wir zuerst den einfachen Fall reeller Werte λ. Mit $\lambda > 0$ ist $h\lambda > 0$ und damit $F_E(\lambda h) > 1$, d.h. die Näherungswerte u_i werden qualitativ richtig berechnet. Dieser Fall ist nicht sehr interessant, weil die in den Naturwissenschaften durch Differentialgleichungen beschriebenen Vorgänge in der Regel exponentiell abklingende Komponenten aufweisen. Dies ist für $\lambda < 0$ der Fall. Die Näherungslösung u_i klingt wie $y(t_i)$ dann und nur dann ab, falls

$$|F_E(\lambda h)| < 1$$

ist; d.h. wenn

$$-2 < h\lambda < 0.$$

Mit anderen Worten, für

$$h > \frac{2}{|\lambda|}$$

erhalten wir ein qualitativ falsches Verhalten (siehe obiges Beispiel!). Die Schrittweite h kann also nicht beliebig gewählt werden.

Systeme von Differentialgleichungen besitzten oft auch oszillierende, exponentiell abklingende Komponenten, welche komplexen Werten von λ entsprechen. Für die dann komplexwertige Lösung $y(x)$ der Anfangswertaufgabe (4.10) gilt wiederum $y(t_{i+1}) = e^{\lambda h} y(t_i)$. Der komplexe Faktor $e^{\lambda h}$ ist im interessanten Fall mit $\mathrm{Re}(\lambda) < 0$ betragsmäßig kleiner Eins. Damit die berechnete Näherungslösung u_i wenigstens wie $y(t_i)$ dem Betrag nach abnehmen, muß wiederum die notwendige und hinreichende Bedingung

$$|F_E(\lambda h)| < 1$$

erfüllt sein.

Definition 4.16 *Für ein Einschrittverfahren, welches für die Testanfangswertaufgabe (4.10) auf* $u_{i+1} = F(h\lambda)u_i$ *führt, heißt die Menge*

$$\mathcal{B} := \{\mu \in \mathbb{C} | \quad |F(\mu)| < 1\}$$

Gebiet der absoluten Stabilität.

Die Schrittweite h ist so zu wählen, daß für $\mathrm{Re}(\lambda) < 0$ stets $h\lambda \in \mathcal{B}$ gilt. Andernfalls liefert das Verfahren unsinnige Ergebnisse, es arbeitet instabil. Diese Stabilitätsbedingung ist speziell bei der Integration von Differentialgleichungssystemen zu beachten, denn die Schrittweite h ist so zu bemessen, daß für alle Abklinkonstanten λ_j mit $\mathrm{Re}(\lambda_j) < 0$ die Bedingung $h\lambda \in \mathcal{B}$ gilt.
Das implizite Eulerverfahren ist ebenfalls ein Einschrittverfahren. Hier folgt aus

$$|F_I(\lambda h)| < 1$$

gerade

$$1 < |1 - h\lambda|.$$

Insbesondere

$$\mathcal{B}_{impl\_euler} = \{\mu \in \mathbb{C} | \quad |\mu - 1| > 1\},$$

siehe Abbildung 4.5. Für das explizite Eulerverfahren gilt

$$\mathcal{B}_{expl\_euler} = \{\mu \in \mathbb{C} | \quad |\mu| < 1\},$$

siehe Abbildung 4.6.
Insbesondere enthält $\mathcal{B}_{impl\_euler}$ die gesamte linke Halbebene; das Verfahren liefert somit qualitativ richtige Ergebnisse für alle λ mit $\mathrm{Re}(\lambda) < 0$. Man bezeichnet Einschrittverfahren, für die

$$\mathcal{B} \subset \mathbb{C}^- = \{\mu \in \mathbb{C} | \quad \mathrm{Re}(\mu) < 0\}$$

als absolut stabil. Das implizite Eulerverfahren ist also absolut stabil.

Bemerkung 4.17 *Ein Verhalten wie in Beispiel 4.2 ii), wo leichte Störungen in den Anfangswerten zu sich exponentiell voneinander entfernten Lösungstrajektorien führen, nennt man inhärente Instabilität. Solche Probleme sind schlecht konditioniert. Hier sollte nur ein Verfahren hoher Fehlerordnung verwendet werden.*

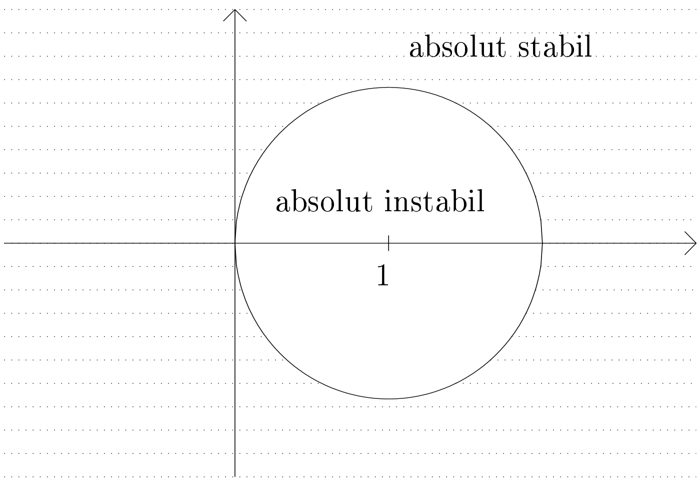


Abbildung 4.5: Bereich absoluter Stabilität beim impliziten Eulerverfahren

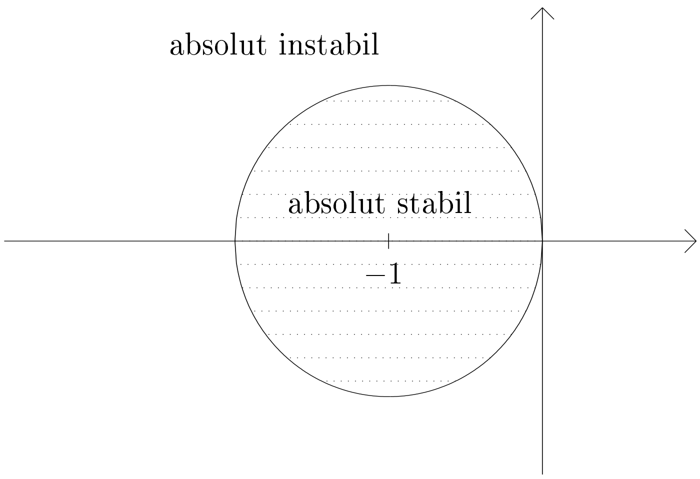


Abbildung 4.6: Bereich absoluter Stabilität beim expliziten Eulerverfahren

4.4 Explizite und implizite Runge-Kutta-Verfahren

Statt $y'(t) = f(t, y(t))$ in den Knoten t_i zu betrachten und dort $y'(t_i)$ geeignet anzunähren, kann man ebensogut

$$\int_{t_j}^{t_{j+1}} y'(s)ds = \int_{t_j}^{t_{j+1}} f(s, y(s))ds$$

betrachten. Hier ist nun das Integral $\int_{t_j}^{t_{j+1}} f(s, y(s))ds$ anzunähern. Für die Entwicklung von Runge-Kutta-Formeln wählt man die Approximation

$$\int_{t_j}^{t_{j+1}} f(s, y(s))ds \approx h \sum_{r=1}^{m} \gamma_r f(s_r, y(s_r))$$

und damit

$$y(t_{j+1}) \approx y(t_j) + h_j \sum_{r=1}^{m} \gamma_r f(s_r, y(s_r)),$$

wobei die s_l m Stützstellen im Intervall $[t_j, t_{j+1}]$ und die γ_l Gewichte der gewählten Quadratur sind und $h_j = t_{j+1} - t_j$. Das Problem besteht hier noch darin, daß in den Ausdrücken $f(s_r, y(s_r))$ die Größe $y(s_r)$ unbekannt ist. Sie muß daher jeweils durch eine geeignete Approximation ersetzt werden.
Man macht dazu folgenden Ansatz, wobei wir wieder $y_j = y(t_j)$ setzen:

$$f(s_r, y(s_r)) \approx k_r(t_j, y_j)$$

mit

$$s_1 = t_j \qquad s_r = t_j + \alpha_r h_j \quad \alpha_r = \sum_{l=1}^{r-1} \beta_{rl}$$

$$\begin{aligned}
k_1(t_j, y_j) &= f(t_j, y_j) \\
k_2(t_j, y_j) &= f(t_j + \alpha_2 h_j, y_j + h_j \beta_{21} k_1(t_j, y_j)) \\
k_3(t_j, y_j) &= f(t_j + \alpha_3 h_j, y_j + h_j(\beta_{31} k_1(t_j, y_j) + \beta_{32} k_2(t_j, y_j))) \\
&\vdots \\
k_m(t_j, y_j) &= f(t_j + \alpha_m h_j, y_j + h_j(\beta_{m1} k_1(t_j, y_j) + \ldots + \beta_{m,m-1} k_{m-1}(t_j, y_j)))
\end{aligned}$$

Mit diesem Ansatz führt unsere Idee auf das folgende Verfahren:

$$\begin{aligned}
u_{j+1} &= u_j + h_j(\gamma_1 k_1(t_j, u_j) + \ldots + \gamma_m k_m(t_j, u_j)) \\
u_0 &= y_0
\end{aligned}$$

bzw. auf die zugehörige Inkrementfunktion:

$$\phi(t, y, h_j, f) = \gamma_1 k_1(t, y) + \ldots + \gamma_m k_m(t, y) \tag{4.11}$$

Ein Verfahren dieser Form heißt explizites m-stufiges Runge-Kutta-Verfahren. Es wird üblicherweise durch eine Tabelle angegeben:

$$\begin{array}{c|cccccc}
0 & & & & & & \\
\alpha_2 & \beta_{21} & & & & & \\
\vdots & \vdots & \ddots & & & & \\
\alpha_r & \beta_{r1} & \cdots & \beta_{r,r-1} & & & \\
\vdots & \vdots & & & \ddots & & \\
\alpha_m & \beta_{m1} & \cdots & \cdots & \cdots & \beta_{m,m-1} & \\
\hline
 & \gamma_1 & \cdots & \cdots & \cdots & \gamma_{m-1} & \gamma_m
\end{array} \tag{4.12}$$

Die α_r charakterisieren die Stützstellen der Quadratur, die γ_r sind ihre Gewichte.
Die Parameter γ_j, β_{rj} und α_j müssen gewissen Bedingungen genügen, damit man ein konsistentes, sinnvolles Verfahren erhält. Zunächst möchte man aus Gründen der numerischen Stabilität, daß $\alpha_r, \beta_{rj}, \gamma_j$ alle nichtnegativ sind. Außerdem soll $k_r(t_j, y_j)$ die Größe $f(s_r, y(s_r))$ approximieren. Also

$$\begin{aligned}
k_r(t_j, y_j) &= f(t_j + \alpha_r h_j, y_j + h_j(\beta_{r1}k_1(t_j, y_j) + \ldots + \beta_{r,r-1}k_{r-1}(t_j, y_j)) \\
&\approx f(t_j + \alpha_r h_j, y(t_j + \alpha_r h_j)) \\
&= f(t_j + \alpha_r h_j, y_j + \alpha_r h_j y'(t_j) + O(h_j^2)).
\end{aligned}$$

Durch Taylorentwicklung und Vergleich findet man, falls mindestens eine $O(h_j^2)-$Approximation gefordert wird, die Bedingung

$$\alpha_r = \beta_{r1} + \ldots + \beta_{r,r-1} \tag{4.13}$$

Schließlich erhält man aus der Konsistenzforderung $\phi(t, y, 0, f) = f(t, y)$ für stetige ϕ aus (4.11) die Bedingung

$$\gamma_1 + \gamma_2 + \ldots + \gamma_m = 1. \tag{4.14}$$

Denn für $h = 0$ gilt $k_l(t_j, y_j) = k_1(t_j, y_j) = f(t_j, y_j)$ für alle $l = 1, \ldots, m$. Wir erhalten damit sofort:

Lemma 4.18 *Ein Runge-Kutta-Verfahren gegeben durch (4.12) mit der Bedingung (4.14) ist konsistent.*

Wir haben die folgenden Spezialfälle:
i) Eulerverfahren
$m = 1 \quad \gamma_1 = 1$
$u_{j+1} = u_j + h_j f(t_j, u_j)$
Ordnung 1

$$\begin{array}{c|c}
0 & \\
\hline
 & 1
\end{array}$$

ii) modifiziertes Eulerverfahren
$m = 2 \quad \gamma_1 = 0 \quad \gamma_2 = 1 \quad \beta_{21} = \alpha_2 = \frac{1}{2}$
$u_{j+1} = u_j + h_j f(t_j + \frac{1}{2}h_j, u_j + \frac{1}{2}h_j f(t_j, u_j))$
Ordnung 2

$$\begin{array}{c|cc}
0 & & \\
\frac{1}{2} & \frac{1}{2} & \\
\hline
 & 0 & 1
\end{array}$$

iii) Verfahren von Heun
$m = 2 \quad \gamma_1 = \gamma_2 = \frac{1}{2} \quad \beta_{21} = \alpha_2 = 1$
$u_{j+1} = u_j + \frac{h_j}{2}(f(t_j, u_j) + f(t_j + h_j, u_j + h_j f(t_j, u_j))$
Ordnung 2

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

iv) Klassisches Runge-Kutta-Verfahren
$m = 4$
Ordnung 4

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

Die Konsistenzordnung der Runge-Kutta-Verfahren ist i.a. schwierig zu bestimmen. Taylorentwicklung mit Koeffizientenvergleich führt auf nichtlineare Bedingungsgleichungen für die Parameter der Tabelle, deren Zahl mit wachsender Ordnung schnell wächst, d.h. die Anzahl der Funktionsauswertungen wächst auch sehr schnell mit wachsendem p (mehr Parameter nötig!)

Ordnung p	1	2	3	4	5	6	7	8
Zahl der Bed.	1	2	4	8	17	37	85	200

Die erreichbare Konsistenzordnung p einer expliziten Runge-Kutta-Formel ist

m	1	2	3	4	5	6	7	8	9	$m \geq 9$
$p(m)$	1	2	3	4	4	5	6	6	7	$\leq m - 2$

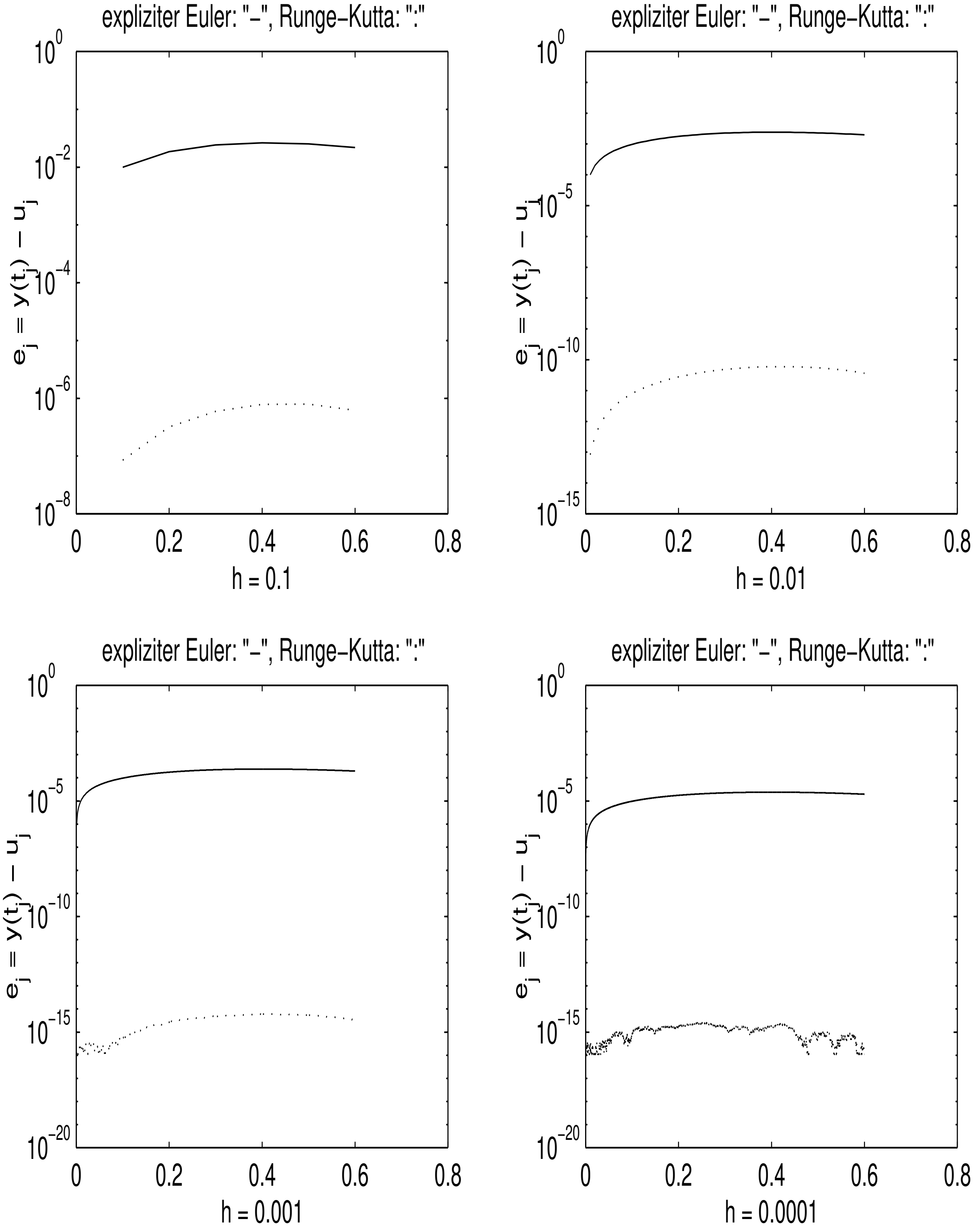
Für Diagramme und Bestimmung der Parameter siehe z.B.Hairer/Nørsett/ Wanner oder direkt bei Butcher, J.C.: On Runge-Kutta Processes of Higher Order, J. Austr. Math. Soc. 4 (1964) 179 - 194 und On The Admissable Order of Runge-Kutta Methods, Math. Comp. 19 (1965) 408 - 417.

Beispiel 4.19 *Vergleich zwischen dem Eulerverfahren und dem klassischen Runge-Kutta-Verfahren an Hand der Anfangswertaufgabe*

$$y' = -2ty^2 \quad y(0) = 1$$

auf $[0, 0.6]$. *Die explizite Lösung lautet* $y(t) = \frac{1}{t^2+1}$. *Für* $h = 0.1$ *ist der Fehler beim Eulerverfahren von der Größenordnung* 10^{-2}, *beim klassischen Runge-Kutta-Verfahren von der Größenordnung* 10^{-7}. *Verkleinert man die Schrittweite, so verkleinert sich der Fehler:*

h	*Euler* *Fehler*	*Runge-Kutta* *Fehler*
0.1	10^{-2}	10^{-7}
0.01	10^{-3}	10^{-10}
0.001	10^{-4}	10^{-15}
0.0001	10^{-5}	$10^{-16} = eps$



Beim Eulerverfahren verkleinert sich der Fehler im selben Maße wie die Schrittweite, beim Runge-Kutta-Verfahren, wie zu erwarten, deutlich schneller.

4.4.1 Implizite Runge-Kutta Formeln

Für eine Reihe von Problemklassen benötigt man Verfahren, die zwar viel höheren Rechenaufwand erfordern aber dafür gewisse Stabilitätseigenschaften (später) haben, dazu gehören die impliziten Runge-Kutta-Verfahren. Sie lassen sich als Erweiterung der expliziten Runge-Kutta-Verfahren ganz einfach durch die folgende Verallgemeinerung der Definition (4.11) entwickeln:

$$k_r(t_j, y_j) = f(t_j + \alpha_r h_j, y_j + h_j(\beta_{r1}k_1 + \ldots + \beta_{rm}k_m)) \quad \text{für} \quad r = 1, \ldots, m \tag{4.15}$$

und damit

$$u_{j+1} = u_j + h_j(\gamma_1 k_1(t_j, u_j) + \ldots + \gamma_m k_m(t_j, u_j))$$

Mit $\beta_{rl} = 0$ für $l \geq r$ ist k_r in (4.15) explizit aus $k_1, \ldots, k_{r-1}$ berechenbar, sonst ist (4.15) implizit in den Unbekannten $k_1, \ldots, k_m$. Man beachte: Nicht die obige Verfahrensgleichung ist eine implizite Gleichung für u_{j+1}, sondern die $k_1, \ldots, k_m$ müssen als Lösung des Gleichungssystems (4.15) z.B. mit dem Newton-Verfahren berechnet werden.
Implizite Runge-Kutta-Verfahren werden üblicherweise durch eine Tabelle angegeben:

$$\begin{array}{c|ccc}
\alpha_1 & \beta_{11} & \ldots & \beta_{1m} \\
\alpha_2 & \beta_{21} & \ldots & \beta_{2m} \\
\vdots & \vdots & & \vdots \\
\alpha_m & \beta_{m1} & \ldots & \beta_{mm} \\
\hline
 & \gamma_1 & \ldots & \gamma_m
\end{array}$$

Bei den impliziten Verfahren sind drei verschiedene Formeltypen von besonderem Interesse.

1. Gauß-Form : Alle Parameter $\alpha_j, \beta_{jl}, \gamma_j$ beliebig wählbar

2. Radau-Form : Entweder $\alpha_1 = \beta_{11} = \beta_{12} = \ldots = \beta_{1m} = 0$ oder $\alpha_m = 1$ und $\beta_{1m} = \beta_{2m} = \ldots = \beta_{mm} = 0$.

3. Lobatto-Form : $\alpha_1 = \beta_{11} = \beta_{12} = \ldots = \beta_{1m} = \beta_{2m} = \ldots = \beta_{mm} = 0, \alpha_m = 1$.

Die Namen der drei Typen von Verfahren leiten sich aus dem Umstand her, daß sie im Spezialfall einer von y unabhängigen Funktion f in die gleichnamigen Quadraturformeln übergehen. Die Radau-Formeln haben den Vorteil, daß entweder k_1 oder k_m explizit berechnet werden kann, bei den Lobatto-Formeln können sogar k_1 und k_m explizit berechnet werden, wodurch die Zahl der in jedem Schritt zu lösenden impliziten Gleichungen verringert wird. Dafür muß man eine geringere Konsistenzordnung bei gleicher Stufenzahl in Kauf nehmen. Sie beträgt in der angegebenen Reihenfolge 1. - 3. der Verfahren $p = 2m, p = 2m - 1, p = 2m - 2$.
Wir geben nun einige spezielle implizite Formeln der genannten Typen an.

Gauß-Form : $m = 1, p = 2$
$$\begin{array}{c|c}
\frac{1}{2} & \frac{1}{2} \\
\hline
 & 1
\end{array}$$

Mit diesen Koeffizienten erhält man das folgende Verfahren :

$$\begin{aligned} k_1 &= f(t_j + \frac{h_j}{2}, u_j + \frac{h_j}{2} k_1) \\ u_{j+1} &= u_j + h_j k_1 \end{aligned}$$

Gauß-Form : $m = 2, p = 4$

$$\begin{array}{c|cc} \frac{(3-\sqrt{3})}{6} & \frac{1}{4} & \frac{(3-2\sqrt{3})}{12} \\ \frac{(3+\sqrt{3})}{6} & \frac{(3+2\sqrt{3})}{12} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Radau-Form : $m = 1, p = 1$

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$$

Bei dieser Formel handelt es sich also gerade um das Eulerverfahren.

Radau-Form : $m = 2, p = 3$

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{2}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array} \qquad \begin{array}{c|cc} \frac{1}{3} & \frac{1}{3} & 0 \\ 1 & 1 & 0 \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array}$$

Lobatto-Form : $m = 2, p = 2$

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Setze man diese Werte ein, so erhält man die (explizite) Formel

$$u_{j+1} = u_j + \frac{h_j}{2}(f(t_j, u_j) + f(t_j + h_j, u_j + f(t_j, u_j))).$$

Lobatto-Form : $m = 3, p = 4$

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\ 1 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

Zur Berechnung der k_r muß ein nichtlineares Gleichungssystem gelöst werden (Newton-Verfahren!). Dazu ist natürlich jeweils zunächst zu klären, ob das System überhaupt eindeutig lösbar ist.

Satz 4.20 *f genüge für alle $(t, y_1), (t, y_2) \in I \times \mathbb{R}^n, I = [t_0, t_0 + a]$*

$$\|f(t, y_1) - f(t, y_2)\| \le L\|y_1 - y_2\|.$$

Dann existiert für alle Schrittweiten h mit

$$q = L \cdot h \cdot max_{j=1,\dots,m}(\sum_{k=1}^{m} |\beta_{jk}|) < 1$$

und alle $(t, y) \in I \times \mathbb{R}^n$ eine eindeutig bestimmte Lösung $k_1(t, y), \dots, k_m(t, y)$ von (4.15).

Beweis: mittels Banachschen Fixpunktsatz.

Bemerkung 4.21 *Um $q < 1$ zu garantieren, muß h genügend klein sein. Das führt oft dazu, daß der erhöhte Rechenaufwand die hohe Konsistenzordnung wieder aufhebt.*

Da der Aufwand, Verfahren höherer Ordnung zu erstellen, recht hoch sein kann, wird man sich überlegen müssen, warum man überhaupt an Verfahren höherer Ordnung interessiert ist. Das Ziel bei der Entwicklung numerischer Verfahren zur Lösung von Differentialgleichung ist es, bei kurzer Rechenzeit eine hohe Genauigkeit zu erreichen. Wir hatten bereits im Satz 3.10 gesehen, daß der Rundungsfehlereinfluß der Verkleinerung der Schrittweite h Grenzen setzt. Eine Erhöhung der Genauigkeit kann man dann nur noch durch eine Erhöhung der Konsistenzordnung erreichen.
Darüberhinaus kann man sich überlegen, daß zum Erreichen einer bestimmten Genauigkeit der Rechenaufwand bei Verfahren höherer Ordnung oft geringer ist als bei Verfahren niedriger Ordnung. Dazu betrachten wir einmal grob die Gesamtrechenzeit Z als Funktion der Konsistenzordnung p des Verfahrens. Nach unseren bisherigen Überlegungen können wir annehmen, daß der Fehler im Endpunkt $t_0 + a$ sich als $\epsilon = Kh^p$ darstellt, wobei K eine Konstante ist und h die Schrittweite. Wir nehmen an, daß für einen Schritt der Rechenaufwand pZ_0 beträgt. Dies ist nicht unrealistisch, denn der Hauptaufwand besteht i.a. in den Funktionsauswertungen, von denen man bei einer Konsistenzordnung p häufig p pro Schritt hat. Gelegentlich aber muß man auch $p^j Z_0$ ansetzen. Bei vorgebener Genauigkeit ϵ erhält man aus $\epsilon = Kh^p$ als benötigte Schrittweite $h = \left(\frac{\epsilon}{K}\right)^{\frac{1}{p}}$. Dann ist aber

$$Z = \frac{a}{h} pZ_0 = apZ_0 \left(\frac{K}{\epsilon}\right)^{\frac{1}{p}}$$

Durch Differenzieren dieser Funktion nach p sieht man, daß die Gesamtrechenzeit minimal wird für $p = [ln\left(\frac{K}{\varepsilon}\right)]$ oder $p = [ln\left(\frac{K}{\varepsilon}\right)] + 1$ ($[x]$ = ganzzahliger Anteil von x). Aus diesem Ergebnis ergibt sich, daß man bei größerere verlangter Genauigkeit, d.h. bei kleinem ϵ, mit entsprechend höherer Konsistenzordnung rechnen sollte, um minimale Gesamtrechenzeit zu erreichen. Eine zu groß gewählte Konsistenzordnung führt allerdings wieder zu einem Anwachsen von Z.

4.5 Schrittweitensteuerung bei Einschrittverfahren

Wir haben gesehen, daß wir die Schrittweite h bei einem Einschrittverfahren nicht zu klein machen können (wegen der Rundungsfehler), aber auch die Ordnung p nicht zu groß, weil dann der Rechenaufwand stark ansteigt. In vielen Problemen ist außerdem ein kleines h oder großes p gar nicht notwendig, weil die Lösung schön glatt verläuft. Es wäre daher ideal, die Schrittweite an das Lösungsverhalten anzupassen, d.h. eine Schrittweitensteuerung vorzunehmen, die in jedem Schritt von einen t_j zum nächsten Knoten t_{j+1} eine neue geeignete Schrittweite h_j bestimmt. Dabei sollte h_j möglichst groß gewählt werden, aber noch so, daß der Fehler garantiert unterhalb einer gewissen Schranke liegt. Da wir den globalen

Fehler nicht kennen, verwenden wir den lokalen Fehler. Die Grundidee ist nun, den lokalen Fehler zu schätzen. Falls diese Schätzung oberhalb einer gegebenen Fehlertoleranz liegt, so wird die Schrittweite verkleinert, falls die Schätzung einen sehr viel kleineren Fehler ergibt, wird sie vergrößert
Das Vorgehen kann man kurz gefaßt so skizzieren:

Ziel Wähle die Schrittweite h_j von t_j zum nächsten Knoten t_{j+1} möglichst groß, aber so, daß der lokale Fehler unterhalb einer gegebenen Toleranz liegt.

Idee Wähle eine Anfangsschrittweite h_{j1} für den Schritt von t_j nach t_{j+1}. Berechne eine Schätzung für den lokalen Fehler. Liegt er unterhalb der gegebenen Fehlertoleranz, so akzeptiere diese Schrittweite als h_j und den damit ermittelten Wert u_{j+1}. Wird der geschätzte Fehler zu groß, so verkleinere die Schrittweite. Wird er viel zu klein, so vergrößere die Schrittweite.

Es gibt zwei Möglichkeiten, den lokalen Fehler zu schätzen. Nehmen wir an, wir wollen mit einem Einschrittverfahren der Ordung p, gegeben durch die Inkrementfunktion $\phi_p(t, u, h, f)$ rechnen.

4.5.1 Fehlerschätzung mit einem Verfahren höherer Ordnung

Verwende eine Inkrementfunktion $\phi_{p+1}(t, u, h, f)$ mit der Ordnung $p + 1$, welche durch möglichst wenig zusätzlichen Rechenaufwand ausgewertet werden kann, für die Schätzung. Für τ_p und τ_{p+1}, die entsprechenden lokalen Diskretisierungsfehler gilt dann:

$$\begin{aligned} \tau_p(t_j, u_j, h_j, f) &= \Delta(t_j, u_j, h_j, f) - \phi_p(t_j, u_j, h_j, f) \\ \tau_{p+1}(t_j, u_j, h_j, f) &= \Delta(t_j, u_j, h_j, f) - \phi_{p+1}(t_j, u_j, h_j, f) \end{aligned}$$

Daraus folgt

$$\begin{aligned} \tau_p &= \phi_{p+1} - \phi_p + \tau_{p+1} \\ &\approx \phi_{p+1} - \phi_p \end{aligned}$$

Dies wird als Schätzung verwendet. Das heißt im Grunde, man tut so, als liefere das Verfahren der Ordnung $p+1$ gegenüber dem Verfahren der Ordnung p die exakten Werte. Man wird dabei versuchen, ein ϕ_{p+1} zu finden, daß die für ϕ_p schon berechneten Funktionsauswertungen benutzen kann und möglichst wenig zusätzliche benötigt. Runge-Kutta-Verfahren, bei denen für die gleichen α_r, β_{rj} mit verschiedenen γ_r zwei unterschiedliche Ordnungen entstehen, nennt man eingebettete Runge-Kutta-Verfahren. Es gibt viele Beispiele, aber nicht alle sind brauchbar.
Wir geben zwei Beispiele an. Das eingebettete Runge-Kutta-Fehlberg-4(5)-Verfahren hat die Form

$$
\begin{array}{c|cccccc}
0 & & & & & & \\
\frac{1}{4} & \frac{1}{4} & & & & & \\
\frac{3}{8} & \frac{3}{32} & \frac{9}{32} & & & & \\
\frac{12}{13} & \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & & & \\
1 & \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & & \\
\frac{1}{2} & -\frac{8}{27} & 2 & -\frac{3544}{2565} & \frac{1859}{4104} & -\frac{11}{40} & \\
\hline
y & \frac{25}{216} & 0 & \frac{1408}{2565} & \frac{2197}{4104} & -\frac{1}{5} & 0 \quad \text{Ordnung 4} \\
\hline
\hat{y} & \frac{16}{135} & 0 & \frac{6656}{12825} & \frac{28561}{56430} & -\frac{9}{50} & \frac{2}{55} \quad \text{Ordnung 5}
\end{array}
$$

Dieses Verfahren braucht keine zusätzlichen Funktionsauswertungen. Eine andere exzellente Methode ist das Verfahren von Dormand-Prince-5(4) (DOPRI5) mit den folgenden Koeffizienten

$$
\begin{array}{c|ccccccc}
0 & & & & & & & \\
\frac{1}{5} & \frac{1}{5} & & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & & & \\
\frac{4}{5} & \frac{44}{45} & -\frac{56}{15} & \frac{32}{9} & & & & \\
\frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & & & \\
1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & & \\
1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & \\
\hline
y_1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0 \\
\hline
\hat{y}_1 & \frac{5179}{57600} & 0 & \frac{7571}{16695} & \frac{393}{640} & -\frac{92097}{339200} & \frac{187}{2100} & \frac{1}{40}
\end{array}
$$

Für Details zu Schrittweitensteuerung bei Runge-Kutta-Verfahren und FORTRAN-Programme siehe E.Hairer, S.P. Nørsett, G. Wanner, Solving Ordinary Differential Equations I, Springer 1993.
Die allgemeine Vorgehensweise läßt sich nun wie in Abbildung 4.7 angegeben skizzieren.

Beispiel 4.22 *Schrittweitensteuerung mit einem Verfahren höherer Ordnung beim Verfahren von Heun:*
Das Verfahren von Heun ($p = 2$) lautet:

$$
\begin{aligned}
u_{j+1} &= u_j + \frac{1}{2}h_j\{k_1 + k_2\}, \\
k_1 &= f(x_j, u_j) \\
k_2 &= f(x_j + h_j, u_j + h_j k_1).
\end{aligned}
$$

Schrittweitensteuerung

Gegeben t_0, u_0, a
Anfangsschrittweite $h < a$
Toleranz τol
Intervall $[t_0, t_0 + a]$
2 Inkrementfunktionen ϕ_p, ϕ_{p+1}, zu Verfahren der Ordnung $p, p+1$

```
i = 0
t = t_0 - h
WHILE t + h < t_0 + a
        t_i = t + h
        berechne φ_1 = φ_p(t_i, u_i, h, f)
        berechne φ_2 = φ_{p+1}(t_i, u_i, h, f)
        berechne τ = |φ_2 - φ_1|
        IF  τ < τol
        THEN IF   τ > τol/100
             THEN u_{i+1} = u_i + hφ_1
                  t = t_i
                  i = i + 1
             ELSE h = 2h
             END IF
        ELSE h = h/2
        END IF
END WHILE
h = t_0 + a - t
u_{i+1} = u_i + hφ_p(t_i, u_i, h, f)
```

Abbildung 4.7: Schrittweitensteuerung

Für die Schätzung brauchen wir ein Verfahren der Ordnung 3, daß k_1, k_2 verwendet. Eine mögliche Lösung lautet

$$\begin{array}{c|ccc} 0 & & & \\ 1 & 1 & & \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \\ \hline & \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \end{array}$$

Also

$$\begin{aligned} \hat{u}_{j+1} &= \hat{u}_j + \frac{1}{6} h_j \{k_1 + k_2 + 4k_3\} \\ k_1 &= f(x_j, \hat{u}_j) \\ k_2 &= f(x_j + h_j, \hat{u}_j + h_j k_1) \\ k_3 &= f(x_j + \frac{1}{2} h_j, \hat{u}_j + \frac{1}{4} h_j (k_1 + k_2)) \end{aligned}$$

Diese beiden Verfahren wollen wir nun zur Lösung von

$$y' = -2ty^2, y(0) = 1$$

auf $[0, 0.3]$ *verwenden. Die explizite Lösung lautet* $y(t) = \frac{1}{t^2+1}$.
Der lokale Diskretisationsfehler soll in jedem Schritt betragsmäßig den Wert $\epsilon = 10^{-6}$ *nicht überschreiten. Fällt die Schätzung des lokalen Diskretisationsfehlers* $d_{k+1}^{(H)} = y(t_{k+1}) - y(t_k) - h\phi(t_k, y(t_k), h, f) \approx h/3\{-k_1 - k_2 + 2k_3\}$ *betragsmäßig zu groß aus, soll die Schrittweite halbiert werden. Sinkt sie unter den Wert* 10^{-7}, *soll die Schrittweite verdoppelt werden. In Tabelle 3 ist der Rechengang zusammengestellt. Gestartet wird mit der Versuchsschrittweite* $h = 0.05$, *die im ersten Integrationsschritt sofort angepaßt werden muß. Mit der Schrittweite* $h = 0.025$ *sind sechs Schritte ausführbar. Da auf Grund der Werte von* k_1, k_2 *und* k_3 *an der Stelle* $t_k = 0.125$ *der Schätzwert für* $d_{k+1}^{(H)}$ *betragsmäßig kleiner als* 10^{-7} *ausfällt, wird die Schrittweite versuchsweise verdoppelt. Doch wird der nächste Schätzwert zu groß, so daß die Schrittweite wieder auf den alten Wert verkleinert werden muß. Das scheinbare Versagen der Strategie hängt mit dem Vorzeichenwechsel des lokalen Diskretisationsfehlers zusammen. Nach diesem Schritt ist eine Integration mit* $h = 0.05$ *erfolgreich, und eine nochmalige Vergrößerung der Schrittweite wäre auf Grund des Schätzwertes* $d_{k+1}^{(H)}$ *an sich möglich, doch erfolgt dies nur nach zwei aufeinanderfolgenden erfolgreichen Integrationsschritten. Bei* $t_k = 0.300$ *ist die Schrittweite zu halbieren. Der globale Fehler* $e_k = |u_k - y(t_k)|$ *wurde mit der exakten Lösung berechnet.*

t_k	u_k	$10^8 e_k$	h	k_1	k_2	k_3	$10^8 d_{k+1}^{(H)}$
0	1.00000	0	0.050	0	−0.1000	−0.0498	416(!)
			0.025	0	−0.0500	−0.0249	26
0.025	0.99937	39	0.025	−0.0499	−0.0996	−0.0747	26
0.050	0.99750	77	0.025	−0.0995	−0.1485	−0.1239	24
0.075	0.99440	114	0.025	−0.1483	−0.1962	−0.1722	21
0.100	0.99009	147	0.025	−0.1960	−0.2426	−0.2193	16
0.125	0.98461	173	0.025	−0.2423	−0.2872	−0.2648	9
0.150	0.97799	192	0.050	−0.2869	−0.3714	−0.3291	130(!)
			0.025	−0.2869	−0.3298	−0.3084	1
0.175	0.97028	199	0.050	−0.3295	−0.4093	−0.3694	9
0.225	0.95181	372	0.050	−0.4076	−0.4771	−0.4425	−301(!)
			0.025	−0.4076	−0.4433	−0.4255	−48
0.250	0.94117	327	0.025	−0.4429	−0.4757	−0.4593	−68
0.275	0.92968	260	0.025	−0.4753	−0.5054	−0.4904	−91
0.300	0.91742	172	0.025	−0.5050	−0.5321	−0.5186	−116(!)

Tabelle 3 : Methode von Heun mit Schrittweitensteuerung

4.5.2 Fehlerschätzung mit zwei verschiedenen Schrittweiten

Wir wollen den lokalen Fehler schätzen und sehen daher unser u_j im Schritt von t_j zum nächsten Knoten als exakt an. Da wir ein Verfahren der Ordnung p benutzen, wissen wir, daß für eine Schrittweite h gilt:

$$u_h(t_j + h) = y(t_j + h) + e_p(t_j + h)h^p + \mathcal{O}(h^{p+1}),$$

mit einer geeigneten Funktion $e_p(t)$, wobei $e_p(t_j) = 0$.
Oft kann man zeigen, daß die Funktion e_p von h unabhängig ist. Dann kann man sie nach h um t_j entwickeln:

$$e_p(t_j + h) = e_p(t_j) + e_p'(t_j)h + \mathcal{O}(h^2) = e_p'(t_j)h + \mathcal{O}(h^2)$$

Man erhält damit

$$u_h(t_j + h) \approx y(t_j + h) + e_p'(t_j)h^{p+1} \tag{4.16}$$

Wenn wir eine Schätzung für $e_p'(t_j)$ bekommen, so können wir berechnen, wie klein h sein muß, damit der lokale Fehler $e_p'(t_j)h^{p+1} \approx \epsilon$ für eine vorgegebene Fehlerschranke ϵ ist.
Eine Schätzung für $e_p'(t_j)$ bekommt man aber folgendermaßen:
Mit dem gewählten Verfahren der Ordnung p bestimmt man mit der Schrittweite H und der Schrittweite $\frac{H}{2}$ Näherungswerte $u_H(t_j + H)$ und $u_{\frac{H}{2}}(t_j + H)$ an $y(t_j + H)$. (Beachte, daß man im zweiten Fall zwei Schritte des Verfahrens benötigt.) Es ist dann

$$\begin{aligned} u_{\frac{H}{2}}(t_j + H) &= y(t_j + H) + e_p(t_j + H)(\frac{H}{2})^p + \mathcal{O}(H^{p+1}) \\ u_H(t_j + H) &= y(t_j + H) + e_p(t_j + H)H^p + \mathcal{O}(H^{p+1}). \end{aligned}$$

Also

$$\frac{u_H(t_j+H) - u_{\frac{H}{2}}(t_j+H)}{2^p - 1} \approx e_p(t_j+H)(\frac{H}{2})^p \approx \frac{H^p}{2^p} e_p'(t_j) H \tag{4.17}$$

Dabei kommt die letzte Gleichung wieder aus der Taylorentwicklung von $e_p(t_j+H)$ um t_j. Aus (4.17) gewinnt man also die Schätzung

$$e_p'(t_j) \approx \frac{2^p}{2^p-1} \frac{1}{H^{p+1}} [u_H(t_j+H) - u_{\frac{H}{2}}(t_j+H)]. \tag{4.18}$$

Hiermit kann man nun wiederum h aus $e_p'(t_j)h^{p+1} \approx \epsilon$ schätzen, denn mit (4.18) ist dann

$$\frac{H}{h} \approx \sqrt[p+1]{\frac{2^p}{2^p-1} \frac{u_H(t_j+H) - u_{\frac{H}{2}}(t_j+H)}{\epsilon}}. \tag{4.19}$$

Praktisch wird man folgendermaßen vorgehen beim Schritt von t_j nach $t_j + H$:
Man wählt eine Schrittweite H und berechnet $u_H(t_j+H)$ und $u_{\frac{H}{2}}(t_j+H)$. Hiermit ermittelt man die Schätzung für $e_p'(t_j)$ nach (4.18) und berechnet mit dieser Schätzung h aus (4.19). Falls $\frac{H}{2} \gg h$, so verwirft man das errechnete Ergebnis, denn der Fehler ist viel größer als das vorgeschriebene ϵ. Es ist daher zweckmäßig H zu reduzieren. Dazu ersetzt man H durch $2h$, berechnet mit dem neuen H wieder $u_H(t_j+H), u_{\frac{H}{2}}(t_j+H)$ und aus (4.19) das zugehörige h, bis schließlich $|\frac{H}{h}| \leq 2$ wird. Sobald dies der Fall ist, akzeptiert man $u_{\frac{H}{2}}(t_0+H)$ als Näherungslösung von $y(t_j+H)$ und geht zu dem nächsten Integrationsschritt über.

Beispiel 4.23 *Betrachtet wurde das Anfangswertproblem*

$$y' = -200ty^2 \quad y(-3) = \frac{1}{901}$$

mit der exakten Lösung $y(t) = \frac{1}{1+100t^2}$. *Mit dem Runge-Kutta-Verfahren und der besprochenen Schrittweitensteuerung wurde ein Näherungswert* u_h *für* $y(0) = 1$ *berechnet. Dabei wurde das Kriterium* $\frac{H}{h} \gg 2$ *durch die Abfrage* $\frac{H}{h} \geq 3$ *ersetzt. 12-stellige Rechnung liefert (hier nur je 4 Stellen angegeben):*

$u_h(0) - y(0)$	*Zahl der benötigten Runge-Kutta-Schritte*	*auftretende kleinste Schrittweite*
$-0.13585 \cdot 10^{-6}$	1476	$0.1226 \cdot 10^{-2}$

Für die feste Schrittweite h liefert das Runge-Kutta-Verfahren:

h	$u_h(0) - y(0)$	*Zahl der Runge-Kutta-Schritte*
$\frac{3}{1476} = 0.2032 \cdot 10^{-2}$	$-0.5594 \cdot 10^{-6}$	1476
$0.1226 \cdot 10^{-2}$	$-0.4052 \cdot 10^{-6}$	$\frac{3}{0.1226 \cdot 10^{-2}} = 2446$

Eine feste Schrittweitenwahl liefert also bei gleichem bzw. größerem Rechenaufwand schlechtere Resultate! Im ersten Fall dürfte die Schrittweite $h = 0.2032 \cdot 10^{-2}$ *in dem „kritischen" Bereich nahe* 0 *zu groß sein; der Diskretisierungsfehler wird zu groß. Die Schrittweite* $h = 0.1226 \cdot 10^{-2}$ *wird dagegen in dem „harmlosen" Bereich von* -3 *bis nahe* 0 *zu klein sein; man macht unnötig viele Schritte und erhält dadurch einen großen Rundungsfehler.*

4.6 Einführung von Mehrschrittverfahren

Ein Nachteil der Einschrittverfahren ist die hohe Anzahl von Funktionsauswertungen, die man für eine hohe Konsistenzordnung benötigt. Die Ordnung eines bestimmten Verfahrens ist fest. Ein Vorteil der Einschrittverfahren besteht darin, daß die Schrittweitensteuerung recht einfach durchzuführen ist.
Im folgenden werden wir Mehrschrittverfahren betrachten, deren Vorteil darin besteht, daß bei geringem Rechenaufwand eine hohe Konsistenzordnung erreichbar ist. Die Ordnung ist für ein gewähltes Verfahren ebenfalls fest, aber Verfahren höherer Ordnung sind leicht konstruierbar. Die Schrittweite sollte nach Möglichkeit konstant sein, sonst ist der Rechenaufwand hoch.
Sei im folgenden $h = a/N$ fest, wobei a die Länge des betrachteten Intervalls sei, und sei $t_j = t_0 + jh, j = 0, \dots, N$. Ein k-Schrittverfahren zur Lösung der Anfangswertaufgabe $y'(t) = f(t, y(t)), y(t_0) = y_0$ ist ein Verfahren zur Berechnung von Näherungswerten u_{j+k} an $y(t_{j+k})$ aus $u_j, \dots, u_{j+k-1}$ durch Lösen der Gleichung

$$\frac{1}{h}\sum_{l=0}^{k} \alpha_l u_{j+l} = \phi(t_j, u_j, \dots, u_{j+k}, h, f) \tag{4.20}$$

für $j = 0, \dots, N-k$ mit geeigneten Startwerten $u_0, \dots, u_{k-1}$. Dabei ist $\frac{1}{h}\sum_{l=0}^{k} \alpha_l u_{j+l}$ als eine Näherung an die Ableitung y' [bisher: $\frac{1}{h}(u_{j+1} - u_j)$] und $\phi(t_j, u_j, \dots, u_{j+k}, h, f)$ als Näherung an $f(t, y(t))$ aufzufassen.
Ein lineares k-Schrittverfahren ist von der Form

$$\sum_{l=0}^{k} \alpha_l u_{j+l} = h\sum_{l=0}^{k} \beta_l f(t_{j+l}, u_{j+l}) \tag{4.21}$$

für $j = 0, \dots, N-k$ mit geeigneten Startwerten $u_0, \dots, u_{k-1}$. Das Verfahren heißt implizit, falls $\beta_k \neq 0$ ist, sonst explizit.
Bei impliziten linearen k-Schrittverfahren muß u_{j+k} aus (4.21) iterativ als Lösung dieser i.a. nichtlinearen Gleichung in u_{j+k} berechnet werden. (4.21) ist eindeutig lösbar, falls f einer Lipschitzbedingung genügt. Sei dazu o.B.d.A. $\alpha_k = 1$. Eine Fixpunktiteration zur Berechnung von u_{j+k} aus (4.21) berechnet Iterierte $u_{j+k}^{(r)}$ gemäß

$$\begin{aligned} u_{j+k}^{(r)} &= -\sum_{l=0}^{k-1} \alpha_l u_{j+l} + h\sum_{l=0}^{k-1} \beta_l f(t_{j+l}, u_{j+l}) + h\beta_k f(t_{j+k}, u_{j+k}^{(r-1)}) \\ &=: \psi(u_{j+k}^{(r-1)}) \end{aligned}$$

Aus der Lipschitzbedingung für f erhalten wir, daß $||\psi(y) - \psi(z)|| \leq |\beta_k|hL||y - z||$ gilt für alle geeigneten y, z. Ist h hinreichend klein, so gilt auch $||\psi(y) - \psi(z)|| < ||y - z||$ und ψ ist eine Kontraktion, so daß der Banachsche Fixpunktsatz die Existenz einer eindeutigen Lösung u_{j+k} von (4.21) sichert, gegen die die obige Iteration konvergiert.

Wir beschäftigen uns zunächst wieder mit der Frage, wann ein solches Mehrschrittverfahren konsistent ist, d.h. wann diese Diskretisierung der Differentialgleichung lokal vernünftig ist. Betrachten wir wieder $y'(t) = f(t, y(t))$ mit $y(\hat{t}) = z$. Wir definieren den lokalen Diskretisierungsfehler in Analogie zu den Einschrittverfahren als den Fehler, den man erhält, wenn man die tatsächliche Lösung in die Verfahrensgleichung einsetzt.
Der lokale Diskretisierungsfehler des Mehrschrittverfahrens (4.20) ist

$$\tau(\hat{t}, z, h, f) = \frac{1}{h}\sum_{l=0}^{k} \alpha_l y_l - \phi(\hat{t}, y_0, \ldots, y_k, h, f),$$

wobei $y_r = y(\hat{t} + rh)$ ist für alle $r = 0, \ldots, k$.
Das Mehrschrittverfahren heißt konsistent, falls für jedes f, dessen partielle Ableitungen auf $[t_0, t_0 + a] \times \mathbb{R}^n$ existieren, stetig und beschränkt sind, gilt

$$\lim_{h \to 0} \tau(\hat{t}, z, h, f) = 0$$

für alle $\hat{t} \in [t_0, t_0 + a]$ und alle $z \in \mathbb{R}^n$. Es heißt konsistent von der Ordnung p, falls $\tau(\hat{t}, z, h, f) = O(h^p)$.

Beispiel 4.24 *Entwicklung eines Mehrschrittverfahrens*
Betrachte die Taylor-Entwicklungen

$$\begin{aligned} y(t+h) &= y(t) + hy'(t) + \frac{h^2}{2}y''(t) + \frac{h^3}{6}y'''(t) + \ldots \\ y(t-h) &= y(t) - hy'(t) + \frac{h^2}{2}y''(t) - \frac{h^3}{6}y'''(t) + \ldots \end{aligned}$$

Durch Subtraktion erhält man:

$$\begin{aligned} y(t+h) - y(t-h) &= 2hy'(t) + O(h^3) \\ &= 2hf(t, y(t)) + O(h^3) \end{aligned}$$

Das hieraus gebildete Verfahren

$$u_{j+2} - u_j = 2hf(t_{j+1}, u_{j+1})$$

ist ein lineares 2*-Schrittverfahren. Seine Konsistenzordnung ist* 2*, denn*

$$\tau(\hat{t}, z, h, f) = \frac{1}{h}[-y(\hat{t}) + y(\hat{t} + 2h)] - 2f(\hat{t} + h, y(\hat{t} + h)) = O(h^2).$$

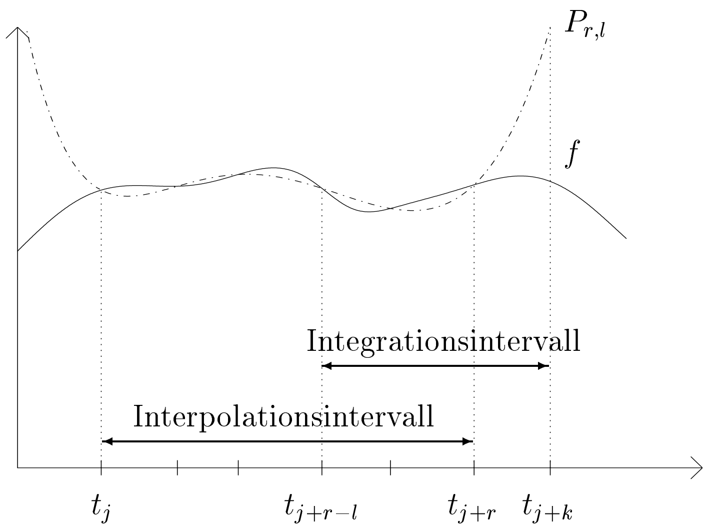
Anders als bei Einschrittverfahren impliziert bei Mehrschrittverfahren die Konsistenz nicht immer Konvergenz. Hierauf werden wir später noch genau eingehen. Zunächst betrachten wir die Entwicklung der wichtigsten Mehrschrittverfahren, die man über numerische Integration herleiten kann wie die Runge-Kutta-Verfahren.

Es gilt

$$y(t_{j+k}) - y(t_{j+r-l}) = \int_{t_{j+r-l}}^{t_{j+k}} f(s, y(s))ds. \tag{4.22}$$

Die Idee ist nun, das Integral näherungsweise zu berechnen. Dazu möchte man hier $f(s, y(s))$ durch das Interpolationspolynom vom Grad r ersetzen, das in den Knoten $t_j, \ldots, t_{j+r}$ gerade den jeweiligen Wert $f(t_j, y(t_j)), \ldots, f(t_{j+r}, y(t_{j+r}))$ annimmt. Wie im Kapitel 3.5 gesegen, ist ein solches Polynom durch die Vorgabe dieser Interpolationsbedingungen eindeutig bestimmt, und es kann einfach berechnet werden. Das Integral über ein gegebenes Polynom ist dann natürlich mittels der leicht auszurechnenden Stammfunktion des Polynoms zu berechnen. Das einzige verbleibende Problem ist hier, daß man die Werte $f(t_j, y(t_j)), \ldots, f(t_{j+r}, y(t_{j+r}))$ nicht kennt. Man ersetzt diese nun wiederum durch die Näherungswerte $f(t_j, u_j), \ldots, f(t_{j+r}, u_{j+r})$.

Das Integral „lebt" auf $[t_{j+r-l}, t_{j+k}]$, während das Polynom, dessen Integral nun hierauf berechnet wird, auf $[t_j, t_{j+r}]$ interpoliert. Also graphisch betrachtet



Das Interpolationspolynom kann man auf verschiedene Weise allgemein darstellen. Eine Möglichkeit ist die Lagrange-Form:

$$P_{r,l}(s) = \sum_{i=0}^{r} f(t_{j+i}, u_{j+i}) \prod_{p=0, p\neq i}^{r} \frac{s - t_{j+p}}{(i-p)h}.$$

Also erhalten wir, falls nun $P_{r,l}(s)$ in (4.22) für f ersetzt wird:

$$y_{j+k} - y_{j+r-l} \approx h \sum_{i=0}^{r} f(t_{j+i}, u_{j+i}) \beta_i^{(r,l)},$$

wobei

$$h\beta_i^{(r,l)} = \int_{t_{j+r-l}}^{t_{j+k}} \prod_{p=0,p\neq i}^{r} \frac{s-t_{j+p}}{(i-p)h} ds = h\int_{-l}^{k-r} \prod_{p=0,p\neq i}^{r} \left(\frac{r-p+\tilde{s}}{i-p}\right) d\tilde{s},$$

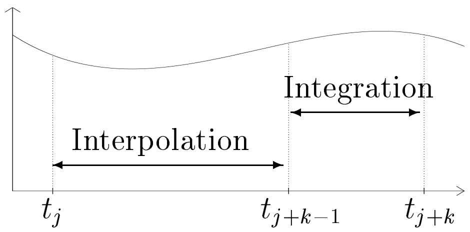
wie man sofort durch Substitution $s = t_j + (r+\tilde{s})h$ ersieht.
Das Verfahren lautet dann also

$$u_{j+k} = u_{j+r-l} + h\sum_{i=0}^{r} \beta_i^{(r,l)} f_{j+i}, \tag{4.23}$$

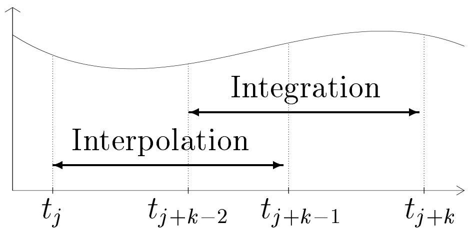
wobei $f_q = f(t_q, u_q)$ ist. Übliche Wahlen für r und l:

r	l	Name	Art
k - 1	0	Adams-Bashforth	explizites Verfahren
k - 1	1	Nyström	explizites Verfahren
k	1	Adams-Moulton	implizites Verfahren
k	2	Milne-Simpson	implizites Verfahren

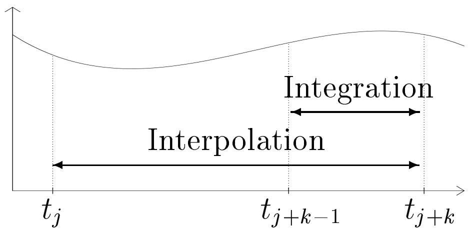
Adams-Bashforth:



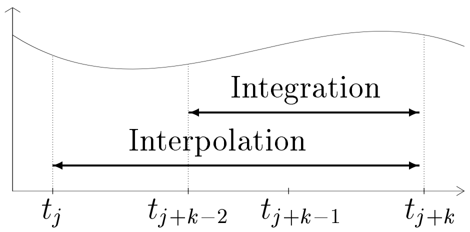
Nyström:



Adams-Moulton:



Milne-Simpson:



Beispiel 4.25

Adams-Bashforth: $u_{j+k} = u_{j+k-1} + h\sum_{i=0}^{k-1} \beta_i^{(k-1,0)} f_{j+i}$

$r = k-1 = 0:$ *Hiermit erhält man:* $u_{j+1} = u_j + hf_j$ *(Euler-Verfahren)*

$r = k-1 = 1:$ *Hiermit erhält man:* $u_{j+2} = u_{j+1} + \frac{h}{2}(3f_{j+1} - f_j)$

Nyström: $u_{j+k} = u_{j+k-2} + h\sum_{i=0}^{k-1} \beta_i^{(k-1,1)} f_{j+i}$

$r = k-1 = 0:$ *Hiermit erhält man:* $u_{j+1} = u_{j-1} + 2hf_j$

Adams-Moulton: $u_{j+k} = u_{j+k-1} + h(\sum_{i=0}^{k-1} \beta_i^{(k,1)} f_{j+i} + \beta_k^{(k,1)} f_{j+k})$

u_{j+k} taucht auf beiden Seiten auf und muß daher durch die Lösung eines nichtlinearen Gleichungssystems iterativ bestimmt werden.

$r = k = 0$ *Hiermit erhält man:* $u_j = u_{j-1} + hf_j$ *(implizierter Euler)*

$r = k = 1:$ *Hiermit erhält man:* $u_{j+1} = u_j + \frac{h}{2}(f_{j+1} + f_j)$ *(Trapezregel)*

$r = k = 2:$ *Hiermit erhält man:* $u_{j+2} = u_{j+1} + \frac{h}{12}(5f_{j+2} + 8f_{j+1} - f_j)$

Milne-Simpson: $u_{j+k} = u_{j+k-2} + h\sum_{i=0}^{k} \beta_i^{(k,2)} f_{j+i}$

$r = k = 2$ *Hiermit erhält man:* $u_{j+2} = u_j + \frac{h}{3}(f_{j+2} + f_{j+1} + f_j)$

Man zeigt: Die so hergeleiteten Mehrschrittverfahren vom Typ (r, l) sind konsistent für $(r+1)$ -mal stetig differenzierbares f, und zwar von der Ordnung $p = r + 1$. Der Beweis erfolgt über die Fehlerbetrachtung bei der Polynominterpolation.

4.7 Konsistenz von Mehrschrittverfahren

Betrachten wir Mehrschrittverfahren wieder allgemein, so stellt man fest, daß bei den Konsistenzüberlegungen zwei Polynome eine wesentliche Rolle übernehmen. Sei also allgemein wieder vorliegend die Verfahrensgleichung

$$\begin{aligned} \frac{1}{h}\sum_{i=0}^{k} \alpha_i u_{j+i} &= \phi(t_j, u_j, \ldots, u_{j+k}, h, f) \\ \text{speziell} &= \sum_{i=0}^{k} \beta_i f_{j+i}. \end{aligned} \tag{4.24}$$

Betrachten wir dazu die Taylorentwicklung bei Einsetzen der Lösung in die Verfahrensgleichung, so erhalten wir aus

$$\begin{aligned} \frac{1}{h}\sum_{i=0}^{k} \alpha_i y(\hat{t}+ih) &= \phi(\hat{t}, y(\hat{t}), \ldots, y(\hat{t}+kh), h, f) + \tau(\hat{t}, y(\hat{t}), h, f) \\ \text{speziell} &= \sum_{i=0}^{k} \beta_i y'(\hat{t}+ih) + \tau(\hat{t}, y(\hat{t}), h, f) \end{aligned}$$

gerade

$$\begin{aligned} \frac{1}{h}\sum_{i=0}^{k} \alpha_i \{y(\hat{t}) + ihy'(\hat{t}) &+ \frac{i^2h^2}{2} y''(\hat{t}) + \ldots\} = \\ &= \phi(\hat{t}, y(\hat{t}), \ldots, y(\hat{t}+kh), h, f) + \tau(\hat{t}, y(\hat{t}), h, f) \qquad (4.25) \\ \text{speziell} &= \sum_{i=0}^{k} \beta_i \{y'(\hat{t}) + ihy''(\hat{t}) + \frac{i^2h^2}{2} y'''(\hat{t}) + \ldots\} + \tau(\ldots). \end{aligned}$$

Möchte man hier, daß $\tau(\hat{t}, y(\hat{t}), h, f) = O(h)$ gilt, so muß offenbar mindestens gelten:

$$\sum_{i=0}^{k} \alpha_i = 0.$$

Im linearen Fall ist dann mit der zusätzlichen Bedingung

$$\sum_{i=0}^{k} \alpha_i i = \sum_{i=0}^{k} \beta_i$$

die Konsistenz gesichert. $\tau(\hat{t}, y(\hat{t}), h, f) = O(h^2)$ erhält man, wenn weiter gilt:

$$\sum_{i=0}^{k} \frac{\alpha_i i^2}{2} = \sum_{i=0}^{k} \beta_i i$$

usw.

Definiert man

$$\rho(z) := \sum_{i=0}^{k} \alpha_i z^i, \qquad \sigma(z) := \sum_{i=0}^{k} \beta_i z^i$$

so gilt:

$$\rho(1) = \sum_{i=0}^{k} \alpha_i, \qquad \rho'(1) = \sum_{i=0}^{k} i\alpha_i, \qquad \sigma(1) = \sum_{i=0}^{k} \beta_i.$$

$\rho(z)$ heißt erstes charakteristisches Polynom des k-Schrittverfahrens (4.24). $\sigma(z)$ heißt zweites charakteristisches Polynom des linearen k-Schrittverfahrens (4.25).
Mit obigen Überlegungen erhält man folgende Ergebnisse:

Lemma 4.26 *Sei $y \in C^1(I), \rho(1) = 0$. Es gelte*

$$\lim_{h\to 0} \max_{t\in I} |\phi(t, y(t), \dots, y(t+kh), h, f) - \rho'(1) f(t, y(t))| = 0.$$

Dann ist das Verfahren (4.24) konsistent.

Lemma 4.27 *Das lineare k-Schrittverfahren (4.24) ist konsistent genau dann, wenn $\rho(1) = 0$ und $\rho'(1) = \sigma(1)$.*

Die Konsistenz ist hier also äquivalent zu einer algebraischen Bedingung. Im linearen Fall gilt dies sogar für höhere Konsistenzordnungen, wie der folgende Satz zeigt.

Satz 4.28 *Das lineare k-Schrittverfahren (4.20) ist konvergent von der Ordnung p genau dann, wenn gilt*

$$\sum_{i=0}^{k} (i^l \alpha_i - l i^{l-1} \beta_i) = 0 \text{ für alle } l = 0, \dots, p.$$

Beispiel 4.29 *Betrachten wir das Adams-Moulton-Verfahren aus Beispiel 4.25. $r = 1, k = 1, l = 1$:*

$$u_{j+1} - u_j = \frac{h}{2}(f_{j+1} + f_j)$$

Hier ist $\rho(z) = \sum_{l=0}^{1} \alpha_l z^l = z - 1$, also $\alpha_1 = 1, \alpha_0 = -1$ und $\sigma(z) = \sum_{l=0}^{1} \beta_l z^l = \frac{1}{2}z - \frac{1}{2}$, also $\beta_0 = \frac{1}{2}, \beta_1 = \frac{1}{2}$. Damit ergibt sich für die Bedingung aus Satz 4.28

$$\begin{aligned}
l = 0: &\quad \sum_{i=0}^{1} \alpha_i = 1 - 1 = 0 \\
l = 1: &\quad \sum_{i=0}^{1} i\alpha_i - \beta_i = -\frac{1}{2} + 1 - \frac{1}{2} = 0 \\
l = 2: &\quad \sum_{i=0}^{1} i^2\alpha_i - 2i\beta_i = 1 - 2 \cdot \frac{1}{2} = 0 \\
l = 3: &\quad \sum_{i=0}^{1} i^3\alpha_i - 3i^2\beta_i = 1 - 3\frac{1}{2} \neq 0.
\end{aligned}$$

Die Ordnung ist also $p = 2$.

Die Bedingungen in Satz 4.28 kann man auffassen als Gleichungen zur Bestimmung geeigneter α_j, β_j, so daß man eine gewünschte Konsistenzordnung hat. Sieht man sich an, wann diese Gleichungen lösbar sind, so führt dies sofort zu folgender Erkenntnis.

Korollar 4.30 *1. Für $0 \leq r \leq k$ existieren eindeutig bestimmte $\rho(z) = \sum_{i=0}^{k} \alpha_i z^i$ mit $\alpha_k = 1$ und $\sigma(z) = \sum_{i=0}^{r} \beta_i z^i$, so daß das zugehörige k-Schrittverfahren die Ordnung $k + r$ hat (d.h. $p = 2k$ ist erreichbar).*

2. Zu vorgegebenem $\rho(z) = \sum_{i=0}^{k} \alpha_i z^i$ mit $\rho(1) = 0$ existiert ein eindeutig bestimmtes $\sigma(z) = \sum_{i=0}^{r} \beta_i z^i, r \leq k$, so daß das zugehörige Verfahren die Ordnung $p = r + 1$ hat.

Die zweite Aussage des Korollars ist wichtig, weil wie wir gleich sehen werden, $\rho(z)$ verantwortlich ist für eine eventuelle Nicht-Konvergenz eines konsistenten Verfahrens. Es gibt für die Konvergenzüberlegung „gute“und „schlechte“$\rho(z)$. Die Aussage (ii) versichert, daß zu jedem guten $\rho(z)$ ein geeignetes $\sigma(z)$ bestimmbar ist, so daß eine erwünschte Ordnung $\leq k$ erreicht wird.

4.8 Prediktor-Korrektor-Verfahren

Bei impliziten Verfahren ist noch zu überlegen, wie man bei der tatsächlichen Rechnung an einem geeigneten Startwert für die Fixpunktiteration gelangt, mit der die Gleichung des impliziten Verfahrens in jedem Schritt gelöst werden muß. Dazu kombiniert man im allgemeinen das implizite Verfahren mit einem passenden expliziten. Die Kombination zwei solcher Verfahren nennt man Prediktor-Korrektor-Verfahren. Dabei dient das explizite Verfahren dazu, einen Näherungswert $u_{j+k}^{(0)}$ für $y(t_{j+k})$ zu verschaffen, der dann als Startnäherung für das iterative Verfahren benutzt wird, das zur Lösung der Gleichung des impliziten Verfahrens in diesem Schritt eingesetzt werden muß. Dabei gibt es verschiedene Möglichkeiten von Kombinationen. Wir wollen dies hier präzisieren und betrachten, welche Konsistenzordnung Prediktor-Korrektor-Verfahren haben.
Gegeben seien ein explizites Verfahren (Prediktor):

$$\sum_{i=0}^{k} \alpha_i^\star u_{j+i} = h \sum_{i=0}^{k-1} \beta_i^\star f_{j+i}, \quad \alpha_k^\star = 1$$

und ein implizites Verfahren (Korrektor):

$$\sum_{i=0}^{k} \alpha_i u_{j+i} = h \sum_{i=0}^{k} \beta_i f_{j+i}, \quad \alpha_k = 1.$$

Dann erhalten wir hiermit das folgende Verfahren:

Prediktor-Korrektor-Verfahren

1. Berechnung der Näherung u_{j+k}
   Sind $u_j, \dots, u_{j+k-1}$ akzeptierte Näherungen, so berechne
   $u_{j+k} = -\sum_{i=0}^{k-1} \alpha_i^\star u_{j+i} + h \sum_{i=0}^{k-1} \beta_i^\star f_{j+i}$ (Anwendung des Prediktors)
2. Setze $u_{j+k}^{(0)} := u_{j+k}$
3. Iteriere gemäß
   $u_{j+k}^{(m+1)} = -\sum_{i=0}^{k-1} \alpha_i u_{j+i} + h \sum_{i=0}^{k-1} \beta_i f_{j+i} + h\beta_k f(t_{j+k}, u_{j+k}^{(m)})$
   für $m = 0, 1, \dots$ (Anwendung des Korrektors)

Im allgemeinen iteriert man nicht bis zur Konvergenz, sondern nur eine feste Anzahl m_0 von Schritten, meist ein oder zwei.

Beispiel 4.31 *Die Abbildung 4.8 zeigt den Fehler, der bei der Lösung von*

$$y' = -2ty^2,\ y(0) = 1$$

im Intervall $[0, 0.6]$ *beim Euler-Verfahren und zwei Prediktor-Korrektor-Verfahren entsteht. Als Prediktor-Verfahren wurde das Euler-Verfahren, als Korrektor-Verfahren die Trapezregel verwendet. P/C 1 verwendet einen Korrektor-Schritt, P/C 2 zwei. Die exakte Lösung ist gegeben durch* $y = (1+t^2)^{-1}$.

Satz 4.32 *Konsistenzordnung von Prediktor-Korrektor-Verfahren*
Ist f *hinreichend oft differenzierbar, genügt es insbesondere einer Lipschitzbedingung bzgl.* y*, dann ist die Ordnung des Verfahrens* $h^{min(p, m_0+p^\star)}$*, wobei* $p^\star$ *und* p *die Konsistenzordnung des Prediktor- bzw. Korrektorverfahrens ist, d.h.*

$$\begin{aligned} \tau^\star(t+kh, y, h, f) &= c_p^\star h^{p^\star} y^{(p^\star+1)}(t) + O(h^{p^\star+1}) \quad \textit{Prediktor} \\ \tau(t+kh, y, h, f) &= c_p h^p y^{(p+1)}(t) + O(h^{(p+1)}) \quad \textit{Korrektor} \end{aligned}$$

und m_0 *gibt die feste Anzahl von Korrektorschritten (Iterationsschritten) an.*

Bemerkung 4.33 *Falls* $m_0 + p^\star > p$*, so ist die Fehlerkonstante die des Korrektors. Es kann also ein Prediktor mit kleinerer Ordnung gewählt und mittels genügender Iteration die Ordnung des Korrektors mit dessen Fehlerkonstante erreicht werden.*

Beispiel 4.34 *Betrachte die Anfangswertaufgabe* $y' = \lambda y, y(0) = 1$*. Als Korrektor wollen wir die Trapezregel*

$$u_{j+1} = u_j + \frac{h}{2}(f_j + f_{j+1})$$

verwenden, ein Verfahren mit Konsistenzordnung $p = 2$*. Als Prediktor verwenden wir das Eulerverfahren*

$$u_{j+1} = u_j + hf_j,$$

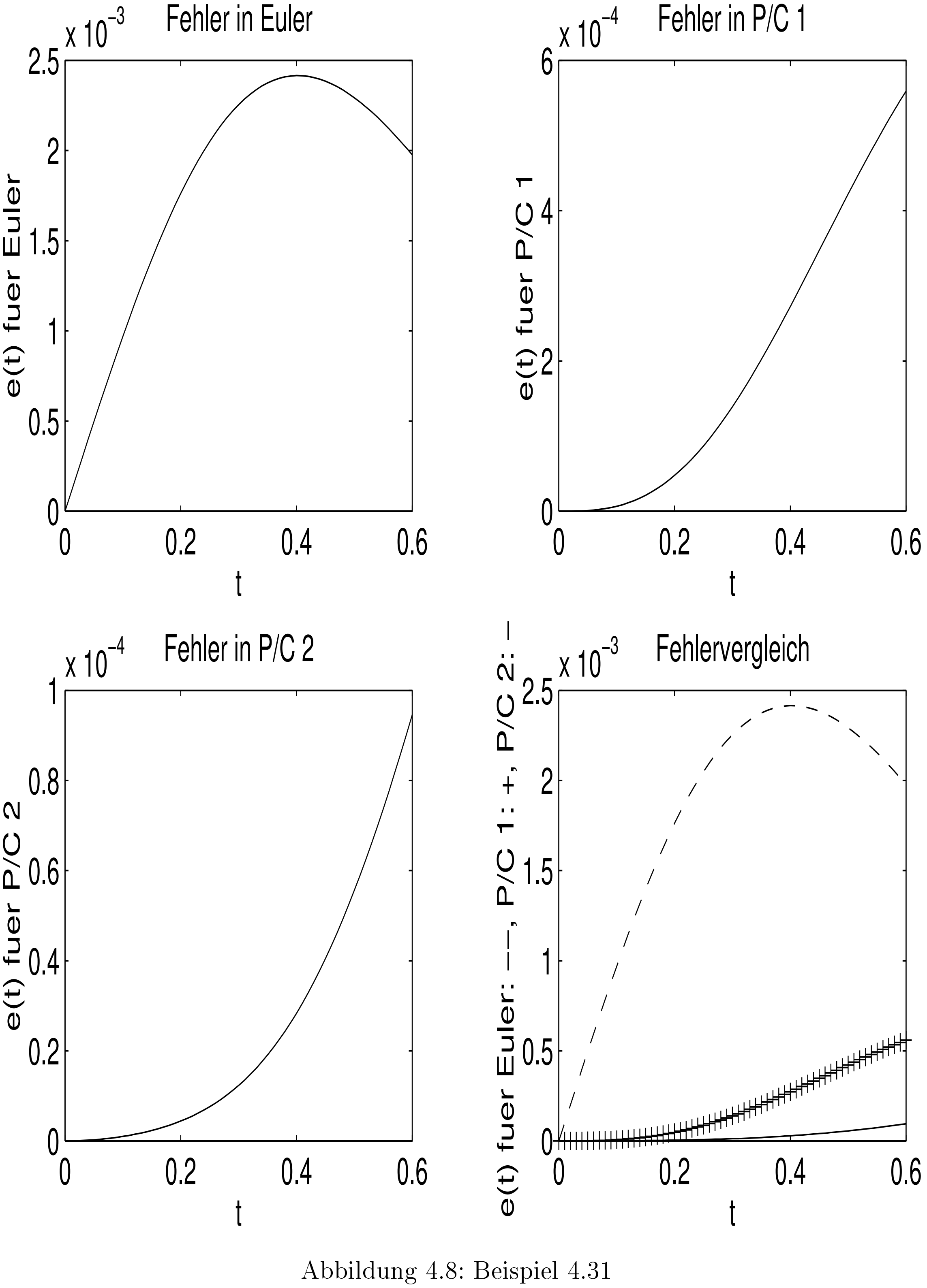


Abbildung 4.8: Beispiel 4.31

dessen Konsistenzordnung $p = 1$ ist. Für obige Anfangswertaufgabe ist $f_j = \lambda u_j$. Mit $q = h\lambda$ erhält man zunächst aus dem Prediktor-Schritt mittels der schon berechneten Lösung $u_j^{(m_0)}$

$$u_{j+1}^{(0)} = (1+q)u_j^{(m_0)}$$

Einsetzen in die Korrektor-Gleichung ergibt

$$\begin{aligned}
u_{j+1}^{(1)} &= u_j^{(m_0)} + \frac{h}{2}f_j^{(m_0)} + \frac{h}{2}f_{j+1}^{(0)} \\
&= u_j^{(m_0)} + \frac{1}{2}qu_j^{(m_0)} + \frac{1}{2}qu_{j+1}^{(0)} \\
&= u_j^{(m_0)} + \frac{1}{2}qu_j^{(m_0)} + \frac{1}{2}q(1+q)u_j^{(m_0)} \\
&= (1+q+\frac{1}{2}q^2)u_j^{(m_0)} \\
u_{j+1}^{(2)} &= (1+q+\frac{q^2}{2}+\frac{q^3}{4})u_j^{(m_0)}
\end{aligned}$$

Die Lösung der Anfangswertaufgabe ist $y(t) = e^{\lambda t}$, so daß für den Diskretisierungsfehler der m_0-fach korrigierten Lösung gilt

$$\begin{aligned}
\tau^{(0)}(t_j, e^{\lambda t_j}, h, \lambda y) &= \frac{1}{h}(e^{\lambda t_{j+1}} - (1+h\lambda)e^{\lambda t_j}) \\
&= \frac{h}{2}\lambda^2 e^{\lambda t_j} + O(h^2),
\end{aligned}$$

da

$$e^{\lambda t_{j+1}} = e^{\lambda t_j} + h\lambda e^{\lambda t_j} + \frac{1}{2}h^2\lambda^2 e^{\lambda t_j} + \frac{h^3\lambda^3}{6}e^{\lambda t_j} + O(h^4).$$

Weiter gilt

$$\begin{aligned}
\tau^{(1)}(t_j, e^{\lambda t_j}, h, \lambda y) &= \frac{1}{h}(e^{\lambda t_{j+1}} - (1+h\lambda+\frac{h^2\lambda^2}{2})e^{\lambda t_j}) \\
&= \frac{h^2\lambda^3}{6}e^{\lambda t_j} + O(h^3) \\
\tau^{(2)}(t_j, e^{\lambda t_j}, h, \lambda y) &= \frac{1}{h}(e^{\lambda t_{j+1}} - (1+h\lambda+\frac{h^2\lambda^2}{2}+\frac{h^3\lambda^3}{4})e^{\lambda t_j}) \\
&= -\frac{h^2\lambda^3}{12}e^{\lambda t_j} + O(h^3)
\end{aligned}$$

Mit der ersten Korrektur erhöht sich die Konsistenzordnung auf die des Korrektorverfahrens $p = 2$. Eine weitere Iteration verringert jetzt nur die Fehlerkonstante, die ab da unverändert bleibt.

Beispiel 4.35 *Prediktor-Korrektor-Verfahren*
Das sogenannte Adams-Bashforth-Moulton-Verfahren, kurz als A-B-M-Methode bezeichnet, ist eine Kombination des impliziten Adams-Moulton-Verfahren mit dem expliziten

Adams-Bashforth-Verfahren. Eine solche Kombination von zwei 3-Schrittverfahren zu einer Prediktor-Korrektor-Methode lautet

$$\boxed{\begin{array}{lcll} u_{k+1}^{(P)} & = & u_k + \frac{h}{12}\{23f_k - 16f_{k-1} + 5f_{k-2}\} & \textit{Adams-Bashforth} \quad O(h^3) \\ u_{k+1} & = & u_k + \frac{h}{24}\{9f(x_{k+1}, u_{k+1}^{(P)}) + 19f_k - 5f_{k-1} + f_{k-2}\} & \textit{Adams-Moulton} \quad O(h^4) \end{array}} \tag{4.26}$$

Als weiteres Beispiel formulieren wir das Prediktor-Korrektor-Verfahren, welches aus der expliziten 4-Schrittmethode von Adams-Bashforth als Prediktor und aus der impliziten 3-Schrittmethode von Adams-Moulton besteht.

$$\boxed{\begin{array}{lcll} u_{k+1}^{(P)} & = & u_k + \frac{h}{24}\{55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3}\} & \textit{Adams-Bashforth} \quad O(h^4) \\ u_{k+1} & = & u_k + \frac{h}{24}\{9f(x_{k+1}, u_{k+1}^{(P)}) + 19f_k - 5f_{k-1} + f_{k-2}\} & \textit{Adams-Moulton} \quad O(h^4) \end{array}} \tag{4.27}$$

Wir behandeln die Anfangswertaufgabe $y' = xe^{x-y}, y(1) = 0$ mit den beiden A-B-M-Methoden 4.25 und 4.26 und wählen die Schrittweite $h = 0.1$. Um die beiden Verfahren unter gleichen Bedingungen vergleichen zu können, werden in beiden Fällen die drei Startwerte u_1, u_2, u_3 mit der klassischen Runge-Kutta-Methode berechnet. Die Tabelle 4.3 enthält die Näherungswerte, die sowohl mit dem Runge-Kutta-Verfahren vierter Ordnung als auch mit den beiden Prädiktor-Korrektor-Methoden erhalten worden sind, wobei für diese Verfahren auch die Prädiktorwerte $u_{k+1}^{(P)}$ angegeben sind. Die Mehrschrittmethoden arbeiten etwas ungenauer im Vergleich zum Runge-Kutta-Verfahren.

4.9 Ein instabiles Mehrschrittverfahren

Bei konsistenten Einschrittverfahren konnte man zeigen, daß sich die lokalen Fehler in der Gesamtrechnung nicht so sehr akkumulieren und in der Folgerechnung auswirken, daß die Konvergenz verhindert wird. Bei Mehrschrittverfahren kann das unter Umständen aber passieren, wie folgendes Beispiel zeigt.

Beispiel 4.36 *Zur Lösung der Anfangswertaufgabe $y' = -y$, $y(0) = 1$ betrachten wir das lineare 2-Schrittverfahren*

$$u_{j+2} + 4u_{j+1} - 5u_j = h(4f_{j+1} + 2f_j)$$

		Runge-Kutta	A-B-M 4.25		A-B-M 4.26	
t_k	$u(t_k)$	u_k	$u_k^{(P)}$	u_k	$u_k^{(P)}$	u_k
1.0	0	0				
1.1	0.2626847	0.26268				
1.2	0.5092384	0.5092357				
1.3	0.7423130	0.7423100				
1.4	0.9639679	0.9639648	0.9641567	0.9639538	0.9638945	0.9639751
1.5	1.1758340	1.1758309	1.1759651	1.1758147	1.1757894	1.1758460
1.6	1.3792243	1.3792213	1.3793138	1.3792027	1.3792028	1.3792382
1.7	1.5752114	1.5752087	1.5752742	1.5751894	1.5751995	1.5752258
1.8	1.7646825	1.7646799	1.7647268	1.7646610	1.7646775	1.7646965
1.9	1.9483792	1.9483769	1.9484106	1.9483588	1.9483785	1.9483924
2.0	2.1269280	2.1269258	2.1269503	2.1269089	2.1269298	2.1269401
⋮	⋮	⋮	⋮	⋮	⋮	⋮
2.5	2.9587436	2.9587422	2.9587472	2.9587315	2.9587478	2.9587508
⋮	⋮	⋮	⋮	⋮	⋮	⋮
3.0	3.7177359	3.7177350	3.7177359	3.7177287	3.7177388	3.7177399

Tabelle 4.3: Runge-Kutta-Methode und A-B-M-Verfahren

mit exakten Anfangswerten $u_0 = 1, u_1 = e^{-h}$. *Das Verfahren ist nach Satz 4.32 von der Ordnung 3. Denn es ist* $\alpha_2 = 1$, $\alpha_1 = 4$, $\alpha_0 = -5$, $\beta_0 = 2$, $\beta_1 = 4$ *und daher*

$$\begin{aligned}
l = 0 &: \sum \alpha_i = 0 \\
l = 1 &: \sum i\alpha_i - \sum \beta_i = 4 + 2 - (4 + 2) = 0 \\
l = 2 &: \sum i^2\alpha_i - \sum 2i\beta_i = 4 + 4 - 2 \cdot 4 = 0 \\
l = 3 &: \sum i^3\alpha_i - \sum 3i^2\beta_i = 4 + 8 - 3 \cdot 4 = 0 \\
l = 4 &: \sum i^4\alpha_i - \sum 4i^3\beta_i = 4 + 16 - 4 \cdot 4 \neq 0.
\end{aligned}$$

Wir rechnen dies einmal mit der Schrittweite $h = 0.1$. *Die Abbildung 4.9 zeigt den Verlauf der Lösungskurve, die sehr harmlos aussieht. Die Abbildung 4.10 zeigt die Näherungskurve. Ab* 0.8 *ist die Näherung überhaupt nicht mehr akzeptabel. Vielleicht ist einfach die Schrittweite zu groß. Rechnen wir noch einmal mit* $h = 0.01$, *so erhalten wir Abbildung 4.11. Beachten Sie die Skalierung!*

Wenn wir die Lösungen betrachten, so sehen wir, daß der Fehler in der Nähe von 1 schlechter wird, wenn h kleiner wird. Dies widerspricht ganz unserer Erfahrung bei Einzelschrittverfahren. Rundungsfehler können dieses Verhalten nicht erklären, denn $h = 0.01$ *ist noch nicht sehr klein. Um zu sehen, was passiert, schauen wir uns das Verfahren genauer an.*

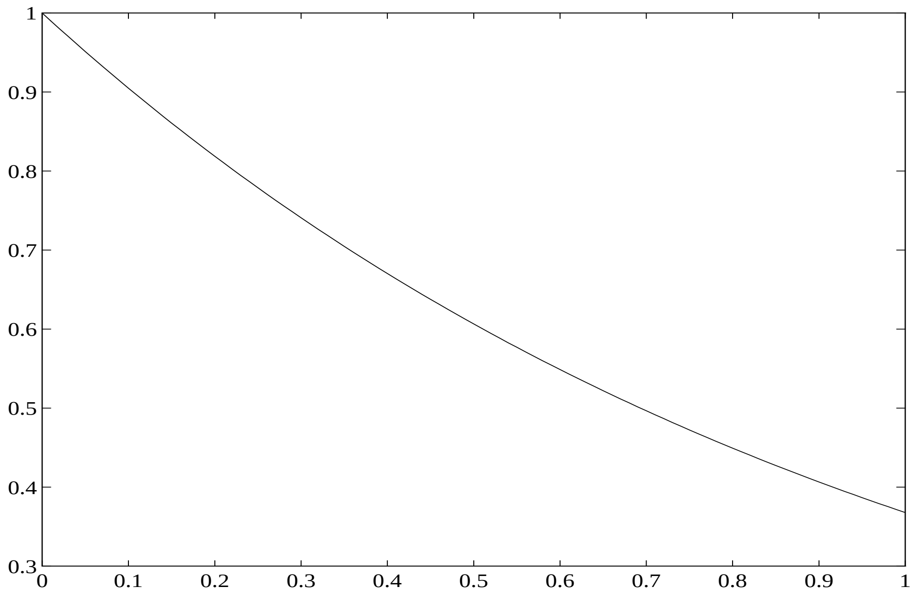


Abbildung 4.9: Verlauf der exakten Lösung

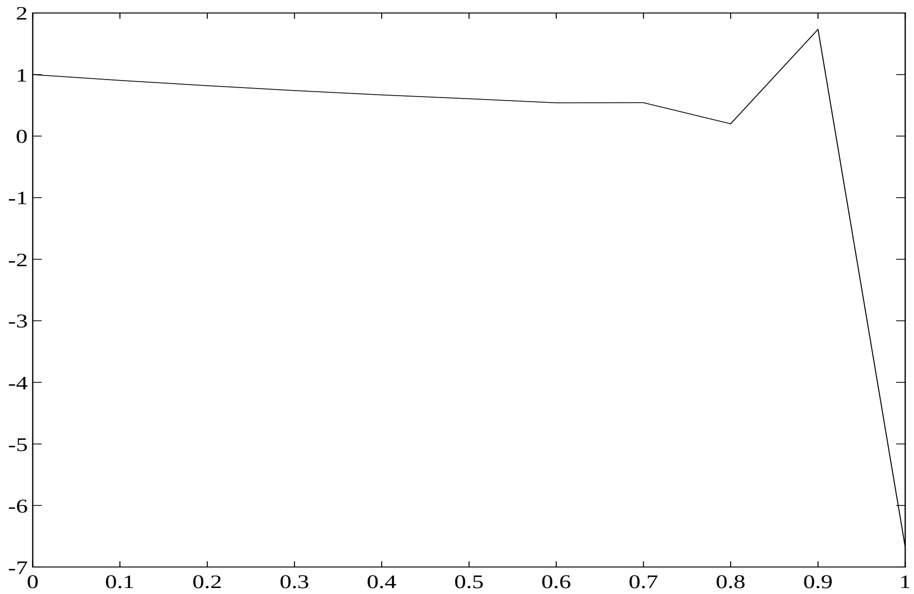


Abbildung 4.10: Verlauf der Näherungskurve $h = 0.1$

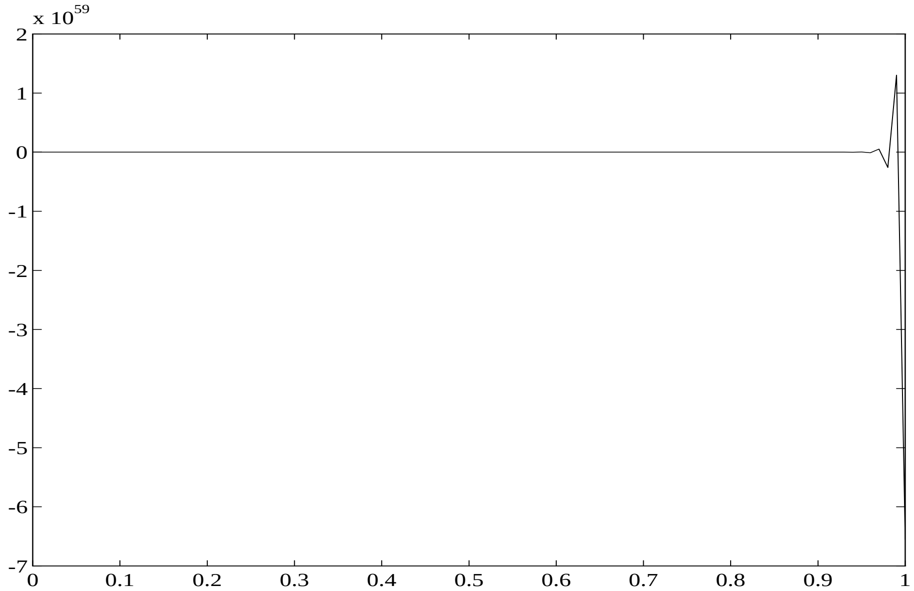


Abbildung 4.11: Verlauf der Näherungskurve $h = 0.01$

Da $f(t,u) = -u$ kann man das Verfahren schreiben als

$$u_{j+2} + (4+4h)u_{j+1} - (5-2h)u_j = 0$$

Das ist eine Differenzengleichung für die Folge $u_0, u_1, u_2, \ldots$.
Aus der Theorie der Differenzengleichung entnimmt man, daß alle Lösungen von der Form $u_l = c_1 z_1^l + c_2 z_2^l$, für alle l, sind. Dabei sind z_1, z_2 die Nullstellen des der Differenzengleichung zugeordneten Polynoms

$$p(z) = z^2 + (4+4h)z - (5-2h) = \rho(z) - h\sigma(z).$$

Dabei sind ρ und σ das erste bzw. zweite charakteristische Polynom des Mehrschrittverfahrens. c_1, c_2 sind Konstanten, die man durch Festlegung der beiden Anfangswerte u_0, u_1 eindeutig bestimmt.

Fragen wir nach dem Konvergenzverhalten, so müssen wir eine feste Stelle $\hat{t} = lh$ mit $l = \frac{\hat{t}}{h}$ betrachten und sehen, was für kleiner werdendes h , d.h. für wachsendes l bei gleichem $\hat{t}$ passiert. Entwickeln wir in u_l die c_1, c_2 und z_1, z_2 nach h, so erhält man nach etlichen Umformungen:

$$u_l = \underbrace{(1 + \mathcal{O}((\frac{\hat{t}}{l})^2))}_{\longrightarrow_{(l\to\infty)} 1} \underbrace{(1 - \frac{\hat{t}}{l} + \mathcal{O}((\frac{\hat{t}}{l})^2))^l}_{\longrightarrow_{(l\to\infty)} e^{-\hat{t}}} - (-\frac{1}{216}(\frac{\hat{t}}{l})^4 + \mathcal{O}((\frac{\hat{t}}{l})^5)(-5 - 3\frac{\hat{t}}{l} + \mathcal{O}((\frac{\hat{t}}{l})^2))^l$$

Der erste Summand approximiert also die Lösung an der Stelle $\hat{t}$. Den zweiten Summanden kann man weiter umformen, so daß man erhält (bis auf Terme höherer Ordnung)

$$-\frac{\hat{t}}{216}\frac{(-5)^l}{l^4}e^{\frac{3\hat{t}}{5}}$$

Für $l \to \infty$ geht dies betraglich gegen ∞ und oszilliert dabei. Verantwortlich für das katastrophale Versagen des Verfahrens ist also die Tatsache, daß $|z_2|^l \to \infty$ für $l \to \infty$.
Beachten Sie , daß für $h \to 0$ das Polynom $p(z)$ gegen $\rho(z) = z^2+4z-5$ mit den Nullstellen $1, -5$ konvergiert.

4.10 Konvergenz von Mehrschrittverfahren

Im letzten Beispiel haben wir gesehen, daß ein konsistentes Mehrschrittverfahren, im Gegensatz zu einem konsistenten Einschrittverfahren nicht konvergieren muss. Unter welchen Bedingungen konvergieren Mehrschrittverfahren?
Betrachten wir in der allgemeinen Situation die MSV-Gleichung für die exakten Lösungswerte $y(t_j)$ und die Näherungswerte $\tilde{u}_j$, wobei wir gleich die Rundungsfehler mit berücksichtigen können, d.h. $\tilde{u}_j$ sind die vom Rechner in unserem Verfahren tatsächlich gelieferten Werte:

$$\begin{aligned}
\sum_{l=0}^{k} \alpha_l y(t_{j+l}) &= h\sum_{l=0}^{k} \beta_l f(t_{j+l}, y(t_{j+l})) + h\tau_{j+k}, \\
\sum_{l=0}^{k} \alpha_l \tilde{u}_{j+l} &= h\sum_{l+0}^{k} \beta_l f(t_{j+l}, \tilde{u}_{j+l}) + h\varepsilon_{j+k}.
\end{aligned}$$

Dabei ist τ_{j+k} der lokale Diskretisierungsfehler bei der Berechnung an der Stelle t_{j+k} und ε_{j+k} ist der Rundungsfehler im j-ten Schritt des Verfahrens. Setzen wir $e_m = y(t_m) - \tilde{u}_m$, so erhalten wir durch Subtraktion der obigen Gleichungen eine Differenzengleichung desselben Typs für die Folge der globalen Fehler $e_0, e_1, \ldots$.

$$\begin{aligned}
\sum_{l=0}^{k} \alpha_l e_{j+l} &= h\sum_{l=0}^{k} \beta_l (f(t_{j+l}, y(t_{j+l})) - f(t_{j+l}, \tilde{u}_{j+l})) + h(\tau_{j+k} - \varepsilon_{j+k}) \\
&= h\sum_{l=0}^{k} \beta_l \frac{\partial f}{\partial y}(t_{j+l}, \eta_{j+l}) e_{j+l} + h(\tau_{j+k} - \varepsilon_{j+k})
\end{aligned}$$

Dabei erhält man die letzte Gleichung nach dem Mittelwertsatz und η_{j+l} ist die entsprechende Zwischenstelle zwischen $y(t_{j+l})$ und $\tilde{u}_{j+l}$.
Für die folgenden Überlegungen machen wir nun zwei Annahmen, nämlich

$$\begin{aligned}
h(\tau_{j+k} - \varepsilon_{j+k}) &= \Phi \text{ konstant}, \\
\frac{\partial f}{\partial y}(t_{j+l}, \eta_{j+l}) &= \lambda \text{ konstant}.
\end{aligned}$$

Die erste Annahme dient dazu, die folgenden Schlüsse einfacher beschreiben zu können. Sie ist i.a. nicht so gravierend, daß sie zu qualitativ anderen Ergebnissen führt. Die zweite kann man auffassen als eine lokale Linearisierung der Differentialgleichung aus der Entwicklung um ein $\hat{y}$:

$$y'(t) = f(t, y(t)) = f(t, \hat{y}) + \frac{\partial f}{\partial y}(t, \hat{y})(y - \hat{y}) + \dots .$$

Wir betrachten dann $y'(t) = f(t, \hat{y}) + \lambda(y - \hat{y})$ und erhalten so

$$\sum_{l=0}^{k} \alpha_l e_{j+l} = h\lambda \sum_{j=1}^{k} \beta_l e_{j+l} + h\Phi.$$

Man kann die Annahme aber auch so interpretieren, daß man sich die Testaufgabe $y'(t) = \lambda y(t)$ anschaut.
Wir untersuchen hier im folgenden nur eine Gleichung, kein System von Differentialgleichungen. Die so gewonnenen Erkenntnisse sind auf Differentialgleichungssysteme sofort übertragbar. Die zweite obige Annahme der Linearising der Differentialgleichung führt für eine System von Differentialgleichungen auf $y'(t) = Ay(t) + b$ mit konstanter Matrix $A \in \mathbb{R}^{n\times n}$. Falls eine nichtsingulären Matrix T existiert, so daß $T^{-1}AT = diag(\lambda_1, \dots, \lambda_n)$ gilt, d.h. falls A diagonalisierbar mit Eigenwerten $\lambda_1, \dots, \lambda_n$ ist, so erhalten wir mit $x(t) = T^{-1}y(t)$ und $c = T^{-1}b$ das transformierte Differentialgleichungssystem

$$\begin{aligned} x'(t) = T^{-1}y'(t) &= T^{-1}ATT^{-1}y(t) + T^{-1}b \\ &= diag(\lambda_1, \dots, \lambda_n)x(t) + c, \end{aligned}$$

d.h. n unabhängige Differentialgleichungen der obigen eindimensionalen Form. Die Untersuchung im Fall $n = 1$ gibt also auch Auskunft über das Verhalten des numerischen Verfahrens im mehrdimensionalen Fall, wenn die beschriebene Transformation möglich ist.
Betrachten wir also nun die Folge der globalen Fehler $e_0, e_1, e_2, \dots$, für die gilt

$$\sum_{l=0}^{k} (\alpha_l - h\beta_l\lambda)e_{j+l} = \Phi \quad \text{für } j = 0, 1, 2, \dots . \tag{4.28}$$

Dies ist eine Differenzengleichung für die Folge $e_0, e_1, e_2, \dots$. Generell gilt für eine Differenzengleichung der Form

$$\sum_{l=0}^{k} \gamma_l e_{j+l} = 0 \quad \text{für alle } j \in \mathbb{N} \tag{4.29}$$

das folgende: Sind $e_0, \dots, e_{k-1}$ gegeben, so sind alle e_m durch (4.29) eindeutig bestimmt. Mit Hilfe des zugeordneten Polynoms

$$q(z) = \sum_{l=0}^{k} \gamma_l z^l$$

kann man die Folge der e_m dann explizit angeben. Im einfachsten Fall hat $q(z)$ paarweise verschiedenen Nullstellen $z_1, \ldots, z_k$. Dann ist

$$e_m = \sum_{p=1}^{k} c_p z_p^m,$$

wobei die Konstanten $c_1, \ldots, c_k$ eindeutig durch die Anfangswerte $e_0, \ldots, e_{k-1}$ bestimmt sind. Hat man statt (4.29) die inhomogene Differenzengleichung

$$\sum_{l=0}^{k} \gamma_l e_{j+l} = \Phi, \tag{4.30}$$

so erhält man jede Lösung als

$$e_m = \hat{e}_m + \eta_m,$$

wobei die Folge $(\hat{e}_m)_{m\in\mathbb{N}}$ irgendeine Lösung der Gleichung (4.30) ist (partikuläre Lösung) und $(\eta_m)_{m\in\mathbb{N}}$ eine Lösung der homogenen Gleichung (4.29) ist. Die dort vorkommenden Konstanten $c_1, \ldots, c_k$ werden den vorgegebenen $e_0, \ldots, e_{k-1}$ angepaßt.
Für unsere Folge der globalen Fehler mit der Differenzengleichung (4.28) ist das zugeordnete Polynom

$$q_h(z) = \sum_{l=0}^{k} (\alpha_l - h\beta_l\lambda) z^l = \rho(z) - h\lambda\sigma(z), \tag{4.31}$$

wobei ρ das erste und σ das zweite charakteristische Polynom des MSVs ist. $q_h(z)$ heißt Stabilitätspolynom des Mehrschrittverfahrens. Falls $\rho(z)$ eine Nullstelle vom Betrag größer als 1 hat oder eine vom Betrag 1 aber mit größerer Vielfachheit als 1, so kann man nicht damit rechnen, daß der globale Fehler e_m gegen 0 konvergiert, wenn h gegen 0 geht. Da für $h \longrightarrow 0$ die Nullstellen z_p von q_h in die Nullstellen von ρ übergehen, müßte im ersten Fall eine Nullstelle $z_p(h\lambda)$ von q_h gegen diese Nullstelle von ρ vom Betrag größer 1 konvergieren und damit muß gelten $|z_p(h\lambda)| > 1$ für alle $h < h_0$, für ein hinreichend kleines h_0. Dann kann e_m wachsen wie $|z_p(h\lambda)|^m$ und ginge damit gegen ∞. Im zweiten Fall hätte man Terme der Form $\mathcal{O}(m|z_p(h\lambda)|^m)$ in der Darstellung von e_m. Man sagt, daß $\rho(z)$ die Wurzelbedingung (auch Stabilitätsbedingung) erfüllt, wenn gilt: Ist $\hat{z}$ eine Nullstelle von $\rho(z)$, so ist entweder $|\hat{z}| < 1$ oder $|\hat{z}| = 1$ und $\hat{z}$ ist eine einfache Nullstelle. Das entsprechende MSV nennt man dann nullstabil. Man beachte, daß bei ESV $\rho(z) = z - 1$. Es gibt hier nur eine Nullstell $\hat{z} = 1$; ESV sind also nullstabil. Man kann zeigen:

Satz 4.37 *Ist das MSV konsistent, so gilt: Das MSV ist konvergent genau dann, wenn die Wurzelbedingung für das erste charakteristische Polynom $\rho(z)$ erfüllt ist. Der globale Fehler ist dann von derselben Ordnung wie der lokale.*

Bemerkung 4.38 *Adams-Bashforth- und Adams-Moulton-Verfahren sind nullstabil, denn ihre ersten charakteristischen Polynome*

$$\rho(z) = z^m - z^{m-1} = z^{m-1}(z-1)$$

haben die einfache Nullstelle $z_1 = 1$ und die $(m-1)$-fache Nullstelle $z_2 = z_3 = \ldots = z_m = 0$. Nyström und Milne-Simpson-Verfahren sind ebefalls nullstabil, denn

$$\rho(z) = z^m - z^{m-2} = z^{m-2}(z^2-1)$$

hat die einfachen Nullstellen $z_1 = 1$, $z_2 = -1$ und die $(m-2)$-fache Nullstelle $z_3 = z_4 = \ldots = z_m = 0$.

Satz 4.39 *(Dahlquist) Für ein lineares k-Schrittverfahren der Ordnung p, daß die Wurzelbedingung erfüllt, gilt stets $p \leq k+2$.*

4.11 Stabilitätskonzepte

Bisher haben wir Aussagen über Konsistenz und Konvergenz hergeleitet, die die Verhältnisse bei gegen Null gehenden Schrittweiten wiedergeben. Jede Rechnung wird jedoch mit einem Schrittweitenvektor $h \neq 0$ durchgeführt, von dem man i.a. nicht weiß, ob er genügend klein ist, daß die asymptotische Theorie das Verhalten der Lösungen (wenigstens) qualitativ richtig beschreibt. Statt an Aussagen für genügend kleines h ist man daher primär an einer Beschreibung des Verhaltens der Näherungslösung für die gerade gewählte Schrittweite interessiert.
Im folgenden setzen wir für unsere MSV stets Konsistenz und Nullstabilität voraus. Wir wissen dann, daß das Verfahren konvergent ist. Wenn wir also eine feste Stelle $\hat{t}$ betrachten und $h \to 0$, so wissen wir, daß die berechneten Näherungswerte $u_{\hat{l}}$ für $y(\hat{t})$ gegen den Lösungswert konvergieren. Die Frage mit der wir uns hier beschäftigen wollen, ist: Können die globalen Fehler $(e_m)_{m\in\mathbb{N}}$ ein Wachstum zeigen, das gute Näherungen verhindert? Mit anderen Worten: können wir für eine feste Schrittweite bei größer werdenden t-Werten Stabilitätsprobleme bekommen?
Im Zusammenhang mit den Einschrittverfahren haben wir schon das Konzept der absoluten Stabilität eingeführt. Das Leitbeispiel war die Differentialgleichung $y' = \lambda y$ mit der Lösung $y(t) = e^{\lambda t}$ und $\lambda < 0$. Die Lösung klingt mit wachsendem t gegen 0 ab und der absolute Fehler sollte dies dann auch tun, denn sonst nähert man für wachsende t die Lösung nicht mehr. Man nennt nicht nur Einschrittverfahren, sondern auch Mehrschrittverfahren absolut stabil, wenn der absolute (globale) Fehler e_m mit wachsendem m kleiner wird. Aufgrund unserer Überlegungen zum globalen Fehler $(e_m)_{m\in\mathbb{N}}$ im letzten Anschnitt können wir sofort sehen, daß ein MSV absolut stabil für $h\lambda$ ist, wenn für die Nullstellen $z_1(h\lambda), \ldots, z_k(h\lambda)$ des Stabilitätspolynoms $q_h(z)$ gilt: $|z_p(h\lambda)| < 1$ für alle $p \in \{1, \ldots, k\}$. Denn wir hatten gesehen, das sich der globale Fehler im einfachsten Falle von paarweise

verschiedenen Nullstellen z_p des Stabilitätspolynoms als

$$e_m = \hat{e}_m + \sum_{p=0}^{k} c_p z_p^m$$

darstellen läßt, wobei $\hat{e_m}$ eine Lösung von (4.29) ist. Sind alle z_p betragsmäßig kleiner als 1, so wird e_m mit wachsendem m kleiner. Die Menge

$$D_\lambda = \{h\lambda | \quad |z_p(h\lambda)| < 1\} \text{ für alle } p \in \{1, \dots, k\}$$

heißt Bereich absoluter Stabilität für das MSV.

Bemerkung 4.40 *Jedes konsistente, nullstabile Verfahren ist für alle hinreichend kleinen positiven λh absolut instabil. In diesem Fall wächst der Fehler, aber auch die Lösung. Der Begriff der absoluten Stabilität ist ein Konzept für $h\lambda$ negativ, d.h. für abklingende Lösungen $e^{\lambda t}$ mit $\lambda < 0$. Solche Anteile sind in vielen Differentialgleichungen zumindest lokal (in der Linearisierung $y' = Ay + f$) enthalten.*

Es stellt sich nun die Frage, ob es immer einen Bereich absoluter Stabilität gibt. Dazu schauen wir uns ein Beispiel an.

Beispiel 4.41 *Wir betrachten die Mittelpunktsregel*

$$u_{j+2} = u_j + 2h\lambda u_{j+1}$$

mit dem Stabilitätspolynom $q_h(z) = z^2 - 2h\lambda z - 1$. Die Nullstellen von q_h sind gegeben durch

$$\begin{aligned} z_1 &= h\lambda + \sqrt{1 + h^2\lambda^2} \\ z_2 &= h\lambda - \sqrt{1 + h^2\lambda^2}. \end{aligned}$$

Für $h\lambda \leq 0$ folgt $|z_2| > 1$, für $h\lambda > 0$ folgt $|z_1| > 1$. Also kann nie $|z_1| < 1$ und $|z_2| < 1$ gelten. Es gibt daher keinen Bereich absoluter Stabilität. Dasselbe folgt für das Milne-Simpson-Verfahren

$$\begin{aligned} u_{j+2} &= u_j + \frac{h}{3}(f_{j+2} + 4f_{j+1} + f_j) \\ &= u_j + h\lambda(\frac{1}{3}u_{j+2} + \frac{4}{3}u_{j+1} + \frac{1}{3}u_j) \end{aligned}$$

mit $q_h(z) = (1 - \frac{h\lambda}{3})z^2 - \frac{4}{3}h\lambda z - (1 + \frac{1}{3}h\lambda)$ und den Nullstellen

$$z_{1/2} = \frac{2h\lambda \pm \sqrt{3(3 + h^2\lambda^2)}}{3 - h\lambda}.$$

Bemerkung 4.42 *1. Man kann zeigen, daß „optimale“ Verfahren, das sind solche mit $p = k + 2$, keine Bereiche absoluter Stabilität besitzen.*

2. Bei impliziten Verfahren sind die Bereiche absoluter Stabilität zum Teil erheblich (bis zu 10 mal) größer als bei expliziten Verfahren.

Es gibt zahlreiche weitere Stabilitätskonzepte, auf die wir hier aus Zeitgründen nicht näher eingehen können. So nennt man ein MSV relativ stabil, wenn der globale Fehler e_m nicht stärker wächst als die Lösung.

4.12 Steife Differentialgleichungen

Probleme der Kontrolltheorie und des Ablaufs chemischer Prozesse zeigen häufig ein Phänomen, wie es im folgenden Beispiel auftritt.

Beispiel 4.43 *(siehe Lambert, Computational Methods in Ordinary Differential Equations, 1973, Seite 230).*

Sei

$$y' = Ay, \qquad y(0) = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

mit

$$A = \begin{bmatrix} -21 & 19 & -20 \\ 19 & -21 & 20 \\ 40 & -40 & -40 \end{bmatrix}.$$

Die Eigenwerte von A lauten $-2, -40+40i, -40-40i$. Die exakte Lösung des Differentialgleichungssystems ist gegeben durch $y^T = [u \ v \ w]$ mit

$$\begin{aligned} u(t) &= \frac{1}{2}e^{-2t} + \frac{1}{2}e^{-40t}(\cos 40t + \sin 40t) \\ v(t) &= \frac{1}{2}e^{-2t} - \frac{1}{2}e^{-40t}(\cos 40t + \sin 40t) \\ w(t) &= -e^{-40t}(\cos 40t - \sin 40t) \end{aligned}$$

Der Lösungsverlauf ist in Abbildung 4.12 wiedergegeben. Ab 0.1 *gibt es keine Schwankungen mehr im Lösungsverlauf. Man möchte daher eine nicht zu kleine Schrittweite bei der numerischen Lösung benutzen. Betrachten wir das Eulerverfahren mit der Schrittweite $h = 0.04$, so erhalten wir für $u(t)$ auf* $[0.1, 0.3]$ *die Näherungen * in der Abbildung 4.13 Hier ist der letzte Näherungswert nicht mehr akzeptabel. Viel schlimmer sieht die Situation aus, wenn wir das größere Intervall* $[0.1, 1]$ *betrachten, siehe Abbildung 4.14. (Beachte die geänderte Skalierung!) Diese Näherung ist offensichtlich völlig sinnlos.*

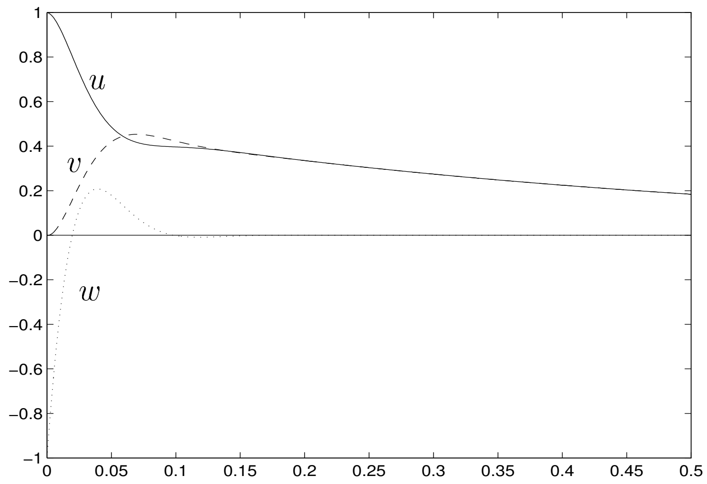


Abbildung 4.12: Lösungsverlauf

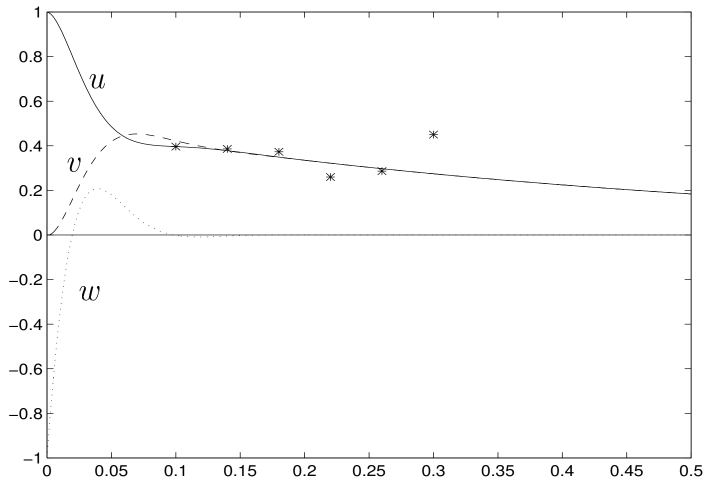


Abbildung 4.13: Euler mit $h = 0.04$, Intervall $[0.1, 0.3]$

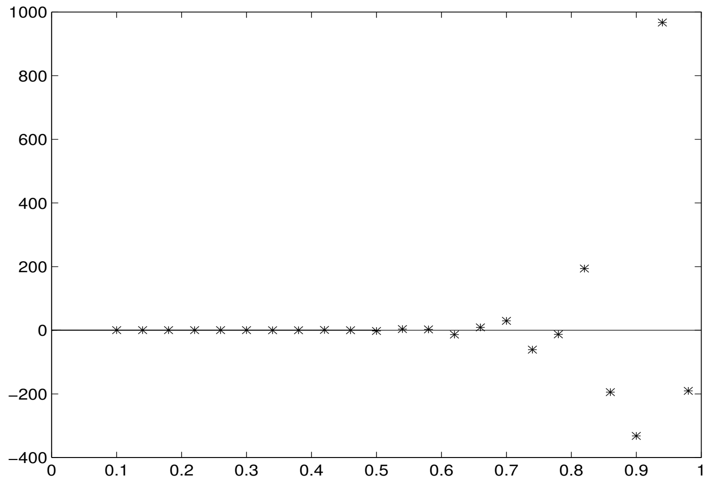


Abbildung 4.14: Euler mit $h = 0.04$, Intervall $[0.1, 1]$

Beispiel 4.44 *Wenn man dagegen das Problem mit*

$$A = \begin{bmatrix} -2 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \; y(0.1) = \begin{bmatrix} \frac{1}{2}e^{-0.2} \\ \frac{1}{2}e^{-0.2} \\ 0 \end{bmatrix}$$

betrachtet, so ist die Lösung gegeben durch

$$y = \begin{bmatrix} \frac{1}{2}e^{-2t} \\ \frac{1}{2}e^{-2t} \\ 0 \end{bmatrix}.$$

Dies ist für $t > 0.1$ nicht von der Lösung aus dem ersten Beispiel zu unterscheiden. Wendet man hierauf das Euler-Verfahren mit Schrittweite $h = 0.04$ an, so erhält man eine befriedigende Näherungslösung.

Der Grund des so unterschiedlichen Verhaltens liegt in den verschiedenen Bereichen der absoluten Stabilität. Der Bereich absoluter Stabilität für das explizite Euler-Verfahren ist, wie wir schon gesehen haben

$$\mathcal{B}_{expl\_euler} = \{\mu \in \mathbb{C} | \quad |\mu| < 1\}$$

(siehe Abbildung 4.6). Damit für das erste Beispiel $h\lambda$ für alle Eigenwerte von A in diesem Bereich liegen, muß $h < 0.025$ sein. Verantwortlich für diese Einschränkung sind die Eigenwerte $-40 \pm 40i$, obwohl ihre Anteile an der Lösung ab 0.1 fast verschwinden. Wählt man h größer, so wächst der absolute Fehler immer stärker. Für das zweite Beispiel sind die Eigenwerte $-2, -2, 0$ und $h < 1$ sichert absolute Stabilität.

Wir betrachten nun allgemein das $n \times n$ System

$$y' = Ay + \phi(t)$$

(z.B. aus der Linearisierung eines Differentialgleichungssystems). A sei diagonalisierbar und $Re(\lambda_i) < 0$ für alle Eigenwerte λ_i von A. Dann hat die Lösung die Form

$$y(t) = \sum_{i=1}^{n} c_i e^{\lambda_i t} v^{(i)} + \psi(t),$$

wobei $v^{(i)}$ der Eigenvektor von A zum Eigenwert λ_i ist. Wegen $Re(\lambda_i) < 0$ geht der erste Teil für $t \to \infty$ gegen 0. Für $t \to \infty$ strebt $y(t)$ also gegen $\psi(t)$. $\psi(t)$ heißt stationäre Lösung. Bei Aufgaben dieser Art ist es das Ziel, diese stationäre Lösung zu bestimmen. Wir müssen dazu die numerische Lösung solange verfolgen, bis der am langsamsten abklingende Term $c_i e^{\lambda_i t} v^{(i)}$ vernachlässigbar klein ist. Falls $|Re(\lambda_i)|$ sehr klein ist, ist dieser Integrationsweg sehr lang. Falls außerdem ein $|Re(\lambda_j)|$ sehr groß ist, so muß für diesen langen Weg eine sehr kleine Schrittweite h verwendet werden, damit $h\lambda_j$ im Bereich absoluter Stabilität liegt. Falls also

$$|Re(\lambda_1)| \geq \ldots \geq |Re(\lambda_n)| \quad \text{und} \quad |Re(\lambda_1)| \gg |Re(\lambda_n)|,$$

so muß sehr lange mit sehr kleiner Schrittweite gerechnet werden. Das hier beschriebene Problem bezeichnet man als das Problem der Steifheit (stiffness, auch Steifigkeit bei manchen Autoren).
Ein System $y' = Ay + \phi(t)$ heißt *steif*, falls

i) $Re(\lambda_i) < 0$ für alle Eigenwerte λ_i von A

ii) $\max_{i=1,\ldots,n} |Re(\lambda_i)| \gg \min_{i=1,\ldots,n} |Re(\lambda_i)|$.

Der Quotient $\frac{\max |Re(\lambda_i)|}{\min |Re(\lambda_i)|}$ heißt Steifigkeitsquotient. Allgemeiner redet man für

$$y' = f(t, y) = f(t, y_0) + \frac{\partial f}{\partial y}(t, y_0)(y - y_0) + \ldots$$

von der Steifheit (auf dem Intervall I), falls die Eigenwerte von $\frac{\partial f}{\partial y}$ *i)* und *ii)* auf I erfüllen.

Bemerkung 4.45 *Im ersten Beispiel dieses Abschnitts ist der Steifigkeitsquotient* 20*, in der Praxis ist* 10^6 *nicht unüblich. Das Problem ist dann,* λh *in den Bereich der absoluten Stabilität zu bekommen.*

Ein Verfahren heißt A-stabil, falls der Bereich absoluter Stabilität die gesamte linke Halbebene umfaßt. Bei einem A-stabilen Verfahren entfallen also die Schwierigkeiten mit steifen Systemen. Man kann folgendes beweisen:

Satz 4.46 *Dahlquist*

i) Ein explizites lineares Mehrschrittverfahren ist niemals A-stabil.

ii) Die Ordnung eines A-stabilen impliziten linearen Mehrschrittverfahrens ist ≤ 2.

iii) Das A-stabile Verfahren 2. Ordnung mit der kleinsten Fehlerkonstante ist die Trapezregel $u_{j+1} - u_j = \frac{h}{2}(f_{j+1} - f_j)$.

Bemerkung 4.47 *Explizite Runge-Kutta-Verfahren haben i.a. einen sehr kleinen Bereich absoluter Stabilität. Implizite Runge-Kutta-Methoden sind A-stabil.*

Beispiel 4.48 *Implizites Runge-Kutta-Verfahren mit* $m = 1$ *für* $y' = \lambda y$.
Wir betrachten das Verfahren

$$u_{j+1} = u_j + k_1$$

mit

$$k_1 = hf(t_j + \frac{1}{2}h, u_j + \frac{1}{2}k_1).$$

In diesem Fall also

$$k_1 = \lambda h(u_j + \frac{1}{2}k_1),$$

so daß gilt

$$(1 - \frac{1}{2}h\lambda)k_1 = h\lambda u_j.$$

Damit ergibt sich

$$u_{j+1} = u_j + \frac{h\lambda}{1 - \frac{1}{2}h\lambda}u_j = \frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda}u_j$$

und

$$q_h(z, h\lambda) = z - \frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda}.$$

Falls $\lambda < 0$, *so ist die Nullstelle stets betraglich* < 1. *Analog für* $m = 2$:

$$u_{j+2} = \frac{1 + \frac{1}{2}h\lambda + \frac{1}{12}h^2\lambda^2}{1 - \frac{1}{2}h\lambda + \frac{1}{12}h^2\lambda^2}\, u_j.$$

Allgemein gilt: Bei expliziten Runge-Kutta-Verfahren sind die Koeffizienten in $q_h(z, h\lambda)$ Polynome in $h\lambda$ (Taylorpolynom von $e^{h\lambda}$). Bei impliziten Runge-Kutta-Verfahren sind sie rationale Funktionen in $h\lambda$ (Padé-Approximation von $e^{h\lambda}$). Diese sind vom Betrag < 1 für $Re(\lambda) < 0$, also sind die Verfahren A-stabil.

Beispiel 4.49 *Explizites Runge-Kutta-Verfahren mit $m = 2$ für $y' = \lambda y$.*
Wir betrachten das modifizierte Euler-Verfahren

$$\begin{aligned} u_{j+1} - u_j &= hf(t_j + \frac{1}{2}h, u_j + \frac{1}{2}hf(t_j, u_j)) \\ &= h\lambda(1 + \frac{1}{2}h\lambda)u_j \\ &= (h\lambda + \frac{h^2\lambda^2}{2})u_j. \end{aligned}$$

Damit ergibt sich

$$q_h(z, h\lambda) = z - (1 + h\lambda + \frac{h^2\lambda^2}{2}).$$

Das Intervall absoluter Stabilität ist daher $]-2, 0[$.