

5 Anfangswertaufgaben gewöhnlicher Differentialgleichungen

5.1 Problemstellung und Beispiele

Bei vielen Prozessen in Natur und Technik wird der zeitliche Ablauf eines Systems dadurch beschrieben, dass eine Beziehung aufgestellt wird zwischen dem Istzustand des Systems und der momentanen Änderung.

BEISPIEL 5.1: (Radioaktiver Zerfall)

Es sei $m(t)$ die Menge radioaktiven Materials zur Zeit t . Der Zerfall des Materials geschieht proportional zur vorhandenen Menge mit dem Proportionalitätsfaktor λ . Somit gilt

$$(5.1) \quad \frac{dm(t)}{dt} = -\lambda m(t) .$$

Dies ist eine gewöhnliche Differentialgleichung. Alle Funktionen $m(t) = Ae^{-\lambda t}$ sind Lösungen dieser Gleichung. Gibt man die zur Zeit t_0 vorhandene Menge m_0 des Materials an, d.h.

$$(5.2) \quad m(t_0) = m_0 ,$$

so erhält man

$$(5.3) \quad m(t) = m_0 e^{-\lambda(t-t_0)} .$$

(5.1) und (5.2) bilden eine sogenannte Anfangswertaufgabe.

Allgemein sieht eine Anfangswertaufgabe wie folgt aus:

Anfangswertaufgabe:

Gesucht ist eine Funktion $y(t)$, so dass gilt

$$(5.4) \quad \frac{dy(t)}{dt} = f(t, y(t)) ,$$

$$(5.5) \quad y(t_0) = y_0 .$$

Hierbei ist der **Anfangswert** y_0 bei $t = t_0$ gegeben. Die **Differentialgleichung** (5.4) ist durch die Angabe der reellen Funktion $f(t, y)$ in den zwei reellen Variablen t und y bestimmt. Im Beispiel 1 ist $f(t, y) = -\lambda y$, wobei λ eine bekannte Zahl ist. f ist in diesem Beispiel in ganz \mathbb{R}^2 definiert. Das braucht nicht notwendigerweise der Fall zu sein. Ist f nur in einem Gebiet $D_f \subset \mathbb{R}^2$ definiert, muss natürlich gelten

$$(t_0, y_0) \in D_f$$

und (5.4) soll nur erfüllt sein, solange

$$(t, y(t)) \in D_f$$

gilt.

Man sagt auch, dass $f(t, y)$ in D_f ein Richtungsfeld bestimme. In jedem Punkt $(t, y) \in D_f$ wird durch eine Gerade mit der Steigung $f(t, y)$ eine Richtung festgelegt. Lösungen der Differentialgleichung (5.4) sind jene Kurven $(t, y(t))$, deren Tangenten in jedem Punkt mit der Geraden des Richtungsfeldes übereinstimmen.

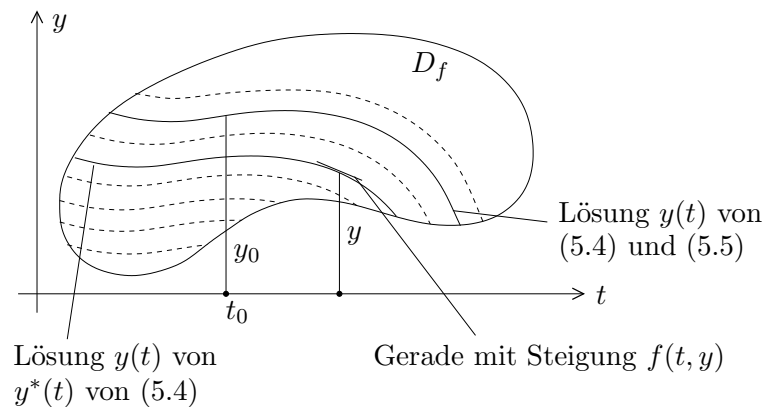


Fig. 5.1

Bei vielen Anwendungsbeispielen lässt sich der Zustand des Systems nicht mit nur einer abhängigen Variablen beschreiben. Wir betrachten die folgenden beiden Beispiele aus der Mechanik.

BEISPIEL 5.2: (Freier Fall)

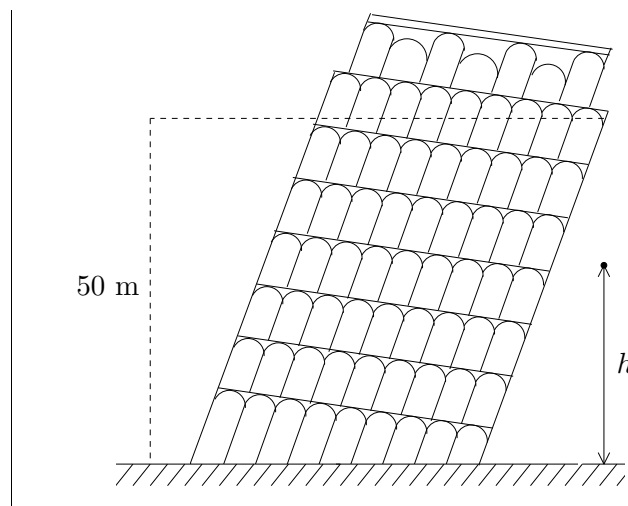


Fig. 5.2

Der Zustand der fallenden Kugel wird beschrieben durch die Höhe $h(t)$ und die Geschwindigkeit $v(t)$ zur Zeit t . Die momentane Änderung der Geschwindigkeit

$$\frac{dv(t)}{dt}$$

ist die Beschleunigung. Sie wird durch das Newtonsche Gesetz (Kraft gleich Masse mal Beschleunigung) gegeben:

$$(5.6) \quad M \frac{dv}{dt} = -Mg$$

($M =$ Masse der Kugel, $g =$ Erdbeschleunigung $\sim 9.81ms^{-2}$). Die momentane Änderung der Höhe $h(t)$ ist die Geschwindigkeit, also

$$(5.7) \quad \frac{dh(t)}{dt} = v .$$

(5.6) und (5.7) bilden ein System von zwei Differentialgleichungen. Dieses hat unendlich viele Lösungen. Gibt man den Anfangszustand an, z.B.

$$(5.8) \quad \begin{aligned} h(0) &= 50 \text{ m} \\ v(0) &= 0 \text{ m/s} , \end{aligned}$$

so ist der freie Fall vollständig beschrieben, und (5.6), (5.7) besitzen genau eine Lösung. Dies ist natürlich ein stark vereinfachtes Modell. Die Gleichungen werden wesentlich komplizierter, wenn man weitere Effekte wie Luftwiderstand, Seitenwind, Erddrehung usw. mitberücksichtigt.

BEISPIEL 5.3: (Erzwungene Schwingung eines stark gedämpften Schwingers)

Es sei eine Masse m an einer als masselos angenommenen Feder aufgehängt. y_0 ist die Ruhelage des Massepunktes, $-c(y - y_0)$ die Rückstellkraft,

$$-\gamma \frac{dy}{dt}$$

die Dämpfungskraft und $P \cos \omega t$ die Störkraft, die mit der Kreisfrequenz ω periodisch das System anregt. Die Differentialgleichung lautet

$$(5.9) \quad m \frac{d^2y}{dt^2} + \gamma \frac{dy}{dt} + c(y - y_0) = P \cos \omega t .$$

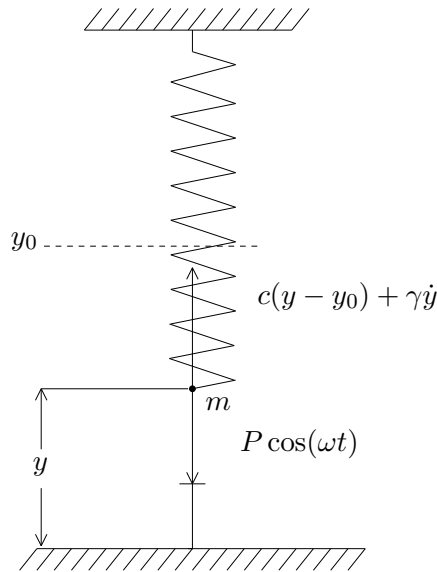


Fig. 5.3

Für spezielle Werte m, γ, c, y_0, P und ω erhält man zum Beispiel

$$(5.10) \quad \frac{d^2 y}{dt^2} + 200 \frac{dy}{dt} + 156.25y = 80 \cos t + 156.25 .$$

Dies ist eine Differentialgleichung zweiter Ordnung. Sie hat genau eine Lösung, wenn man den Anfangszustand festlegt, z.B.

$$(5.11) \quad \begin{aligned} y(0) &= 5 \\ y'(0) &= -100 \end{aligned} .$$

Wir können diese Anfangswertaufgaben (5.10), (5.11) auch als System von Differentialgleichungen schreiben, indem wir die abhängigen Variablen $y_1 := y$ und $y_2 := y'$ einführen. Damit erhalten (5.10) und (5.11) die Form

$$(5.12) \quad \begin{aligned} \frac{dy_1}{dt} &= y_2(t) , \\ \frac{dy_2}{dt} &= -156.25y_1(t) - 200y_2(t) + 80 \cos t + 156.25 , \end{aligned}$$

$$(5.13) \quad y_1(0) = 5, y_2'(0) = -100 .$$

Allgemein sieht eine Anfangswertaufgabe für ein System von m Differentialgleichungen wie folgt aus:

Anfangswertaufgabe für Systeme von m Differentialgleichungen*Gegeben:*

a) m Funktionen $f_1(t, y_1, y_2, \dots, y_m), f_2(t, y_1, \dots, y_m), \dots, f_m(t, y_1, \dots, y_m)$, die in einem Gebiet $D_f \subset \mathbb{R}^{m+1}$ definiert sind.

b) m Anfangswerte $y_{01}, y_{02}, \dots, y_{0m}$ mit t_0 , so dass gilt:

$$(t_0, y_{01}, y_{02}, \dots, y_{0m}) \in D_f.$$

Gesucht:

m Funktionen $y_1(t), y_2(t), \dots, y_m(t)$ so, dass die m Differentialgleichungen

$$(5.14) \quad \begin{cases} \frac{dy_1(t)}{dt} = f_1(t, y_1(t), y_2(t), \dots, y_m(t)) \\ \frac{dy_2(t)}{dt} = f_2(t, y_1(t), y_2(t), \dots, y_m(t)) \\ \vdots \\ \frac{dy_m(t)}{dt} = f_m(t, y_1(t), y_2(t), \dots, y_m(t)) \end{cases}$$

erfüllt sind, solange $(t, y_1(t), y_2(t), \dots, y_m(t)) \in D_f$ gilt. Zudem sollen die Anfangsbedingungen

$$(5.15) \quad \text{Anfangsbedingungen} \quad \begin{cases} y_1(t_0) = y_{01} \\ y_2(t_0) = y_{02} \\ \vdots \\ y_m(t_0) = y_{0m} \end{cases}$$

erfüllt sein.

In Beispiel 2 ist $m = 2$ und

$$y_1 := h, y_2 := v, f_1(t, y_1, y_2) = y_2, f_2(t, y_1, y_2) = -g .$$

In Beispiel 3 ist $m = 2$ und nach (5.12)

$$f_1(t, y_1, y_2) = y_2, f_2(t, y_1, y_2) = -156.25y_1 - 200y_2 + 80 \cos t + 156.25 .$$

Die Anfangswertaufgabe (5.14), (5.15) lässt sich in Vektorschreibweise kürzer schreiben. Sei $\mathbf{y} \in \mathbb{R}^m$ und $\mathbf{f}(t, \mathbf{y})$ eine vektorwertige Funktion

$$\mathbf{f} : \begin{array}{ll} D_f \subset \mathbb{R}^{m+1} & \rightarrow \mathbb{R}^m \\ (t, \mathbf{y}) & \rightarrow \mathbf{f}(t, \mathbf{y}) . \end{array}$$

Die Vektoren $\mathbf{f}(t, \mathbf{y})$, $\mathbf{y}(t)$ und \mathbf{y}_0 haben die explizite Form

$$\mathbf{f}(t, \mathbf{y}) = \begin{pmatrix} f_1(t, y_1, y_2, \dots, y_m) \\ f_2(t, y_1, y_2, \dots, y_m) \\ \vdots \\ f_m(t, y_1, y_2, \dots, y_m) \end{pmatrix},$$

$$\mathbf{y}(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_m(t) \end{pmatrix}, \quad \mathbf{y}_0 = \begin{pmatrix} y_{01} \\ y_{02} \\ \vdots \\ y_{0m} \end{pmatrix}.$$

Unter $d\mathbf{y}/dt$ verstehen wir den durch komponentenweises Differenzieren entstandenen Vektor, d.h.

$$\frac{d\mathbf{y}(t)}{dt} = \begin{pmatrix} \frac{dy_1(t)}{dt} \\ \frac{dy_2(t)}{dt} \\ \vdots \\ \frac{dy_m(t)}{dt} \end{pmatrix}.$$

Damit lassen sich (5.14) und (5.15) schreiben als

$$(5.16) \quad \frac{d\mathbf{y}(t)}{dt} = \mathbf{f}(t, \mathbf{y}(t)),$$

$$(5.17) \quad \mathbf{y}(t_0) = \mathbf{y}_0.$$

In Beispiel 3 waren wir von einer Differentialgleichung zweiter Ordnung ausgegangen und hatten diese in ein System erster Ordnung umgewandelt. Dies geht ganz allgemein bei Differentialgleichungen höherer Ordnung.

A. Anfangswertaufgaben von Differentialgleichungen m -ter Ordnung

Gegeben:

- Eine reelle Funktion $F(t, x_1, x_2, \dots, x_m)$, die in einem $D_F \subset \mathbb{R}^{m+1}$ definiert ist;
- die Anfangswerte $y_{01}, y_{02}, \dots, y_{0m}$ und t_0 , so dass gilt

$$(t_0, y_{01}, y_{02}, \dots, y_{0m}) \in D_F.$$

Gesucht :

Eine m -mal differenzierbare Funktion $y(t)$ so, dass die Differentialgleichung m -ter Ordnung

$$(5.18) \quad \frac{dy^m(t)}{dt^m} = F(t, y(t), y'(t), y''(t), \dots, y^{(m-1)}(t))$$

erfüllt ist, solange $(t, y(t), y'(t), y''(t), \dots, y^{(m-1)}(t))$ in D_F liegt. Zudem sollen die Anfangsbedingungen

$$(5.19) \quad \begin{cases} y(t_0) & = & y_{01} \\ y'(t_0) & = & y_{02} \\ y''(t_0) & = & y_{03} \\ & \vdots & \\ y^{(m-1)}(t_0) & = & y_{0m} \end{cases}$$

erfüllt sein.

Man nennt (5.18) eine Differentialgleichung m -ter Ordnung. Wir können sie sofort umschreiben in ein System erster Ordnung, indem wir die folgenden neuen abhängigen Variablen einführen:

$$\begin{aligned} y_1(t) & := y(t) \\ y_2(t) & := y'(t) \\ & \vdots \\ y_m(t) & := y^{(m-1)}(t) . \end{aligned}$$

Dies führt auf das Differentialgleichungssystem (5.14) mit

$$\begin{aligned} f_1(t, y_1, y_2, \dots, y_m) & := y_2 \\ f_2(t, y_1, y_2, \dots, y_m) & := y_3 \\ & \vdots \\ f_{m-1}(t, y_1, y_2, \dots, y_m) & := y_m \\ f_m(t, y_1, y_2, \dots, y_m) & := F(t, y_1, y_2, \dots, y_m) . \end{aligned}$$

Die Anfangsbedingungen (5.19) gehen hierbei direkt in (5.15) über. Da diese Umformung auf ein System erster Ordnung stets möglich ist, wollen wir uns in diesem Kapitel auf die numerische Lösung von Systemen von Differentialgleichungen erster Ordnung beschränken. (Es gibt aber eine Vielzahl von Verfahren zum direkten Lösen von Differentialgleichungen zweiter Ordnung.)

Satz 5.1. (*Existenz*)

Es sei $\mathbf{f}(t, \mathbf{y}) : D_f \subset \mathbb{R}^{m+1} \rightarrow \mathbb{R}^m$ eine stetige Funktion. Dann besitzt die Anfangswertaufgabe (5.16), (5.17) eine Lösung, die bis an den Rand von D_f fortgesetzt werden kann.

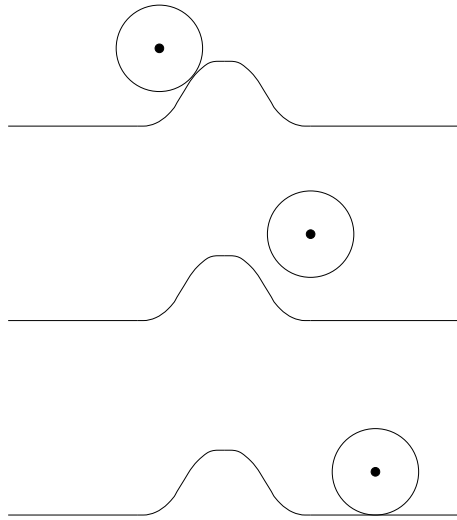


Fig. 5.4

Selbst wenn die rechte Seite der Differentialgleichung unstetig ist, kann man oft eine sinnvolle Lösung angeben. Als Beispiel denke man an das Verhalten eines Automobils beim schnellen Überfahren einer Bodenwelle. Hebt kurzfristig ein Rad vom Boden ab, so besitzt die rechte Seite eine Unstetigkeit beim Wiederaufprall auf die Fahrbahn. Numerisch geht man so vor, dass man die Unstetigkeitsstelle numerisch bestimmt und nach dieser Unstetigkeitsstelle neu zu integrieren beginnt. Wir wollen deshalb annehmen, dass $\mathbf{f}(t, \mathbf{y})$ stetig ist.

Satz 5.2. (*Eindeutigkeit*)

Es sei $\mathbf{f}(t, \mathbf{y}) : D_f \subset \mathbb{R}^{m+1} \rightarrow \mathbb{R}^m$ stetig in D_f , und $\mathbf{f}(t, \mathbf{y})$ erfülle eine Lipschitzbedingung bezüglich \mathbf{y} , d.h. es gibt ein $L > 0$, so dass

$$\|\mathbf{f}(t, \mathbf{y}) - \mathbf{f}(t, \tilde{\mathbf{y}})\| \leq L\|\mathbf{y} - \tilde{\mathbf{y}}\|$$

für alle $(t, \mathbf{y}) \in D_f$ und $(t, \tilde{\mathbf{y}}) \in D_f$. Dann hat die Anfangswertaufgabe (5.16), (5.17) genau eine Lösung.

Formal sieht das Anfangswertproblem (5.16), (5.17) für ein System genauso aus wie (5.4), (5.5) für eine einzige Gleichung. Dasselbe gilt für die hier vorgestellten Verfahren. Der Einfachheit halber werden wir uns im folgenden auf eine skalare Gleichung (5.4) beschränken. Die Formeln können dann sofort auch auf Systeme angewendet werden. Im Falle einer Gleichung lautet der Satz 5.2

Satz 5.3. (Eindeutigkeit)

Es sei $f(t, y) : D_f \subset \mathbb{R}^2 \rightarrow \mathbb{R}$ stetig in D_f , und $f(t, y)$ erfülle eine Lipschitzbedingung bezüglich y , d.h. es gibt ein $L > 0$, so dass

$$|f(t, y) - f(t, \tilde{y})| \leq L|y - \tilde{y}|$$

für alle $(t, y) \in D_f$ und alle $(t, \tilde{y}) \in D_f$. Dann hat (5.4), (5.5) genau eine Lösung.

Zusatz:

Ist D_f ein Streifen $D_f = S[a, b] = [a, b] \times \mathbb{R}$, so hat (5.4), (5.5) genau eine Lösung, sofern $t_0 \in [a, b]$ liegt und die Lösung existiert für alle t in $[a, b]$.

5.2 Das Euler-Verfahren**A. Diskretisation**

Wir lösen die Anfangswertaufgabe

$$(5.20) \quad y'(t) = f(t, y(t)), t \in [a, b],$$

$$(5.21) \quad y(a) = y_0.$$

Wir führen die äquidistanten Gitterpunkte

$$t_i = a + ih, i = 0, 1, \dots, N, \quad \text{mit} \\ h = \frac{b-a}{N} \quad (\text{Schrittweite})$$

ein.

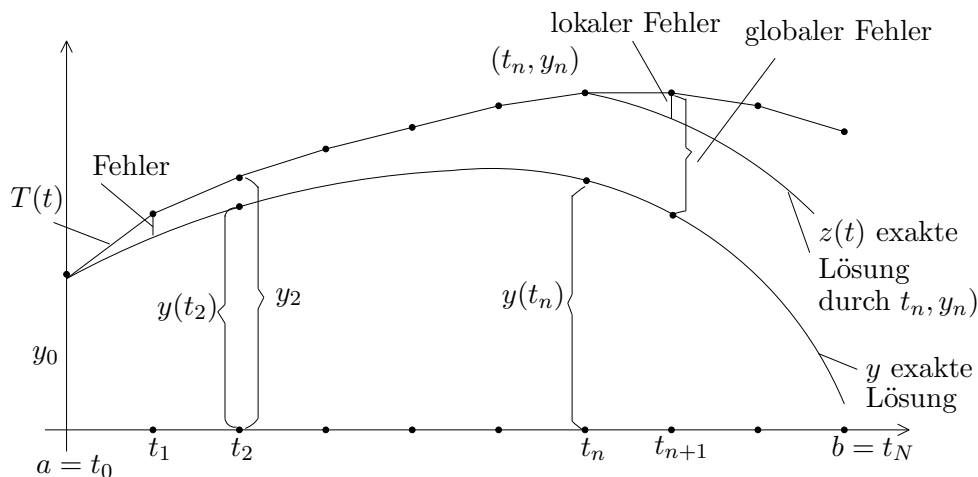


Fig. 5.5

Die exakte Lösung $y(t_i)$ wird durch numerische Werte y_i approximiert.

Ein einfaches Verfahren erhalten wir wie folgt: In jedem Punkt ist das Richtungsfeld tangential zur Lösungskurve. Wir approximieren deshalb wie folgt:

$$y(t_1) = y(a + h) \sim y_1 = y_0 + hy'_0 = y_0 + hf(t_0, y_0) .$$

Das heisst: Ist $T(t)$ die Tangente an $y(t)$ bei $t = a$, so ist $y_1 = T(a + h)$. Da $y(t_1)$ nicht bekannt ist, muss man beim nächsten Schritt von (t_1, y_1) ausgehen, also

$$y(t_2) \sim y_2 := y_1 + hf(t_1, y_1) .$$

Fortsetzen dieser Idee führt auf das

Euler-Verfahren

Gegeben: y_0 Anfangswerte

Für $n = 1, 2, \dots, N$ berechnet man

$$(5.22) \quad y_{n+1} = y_n + hf(t_n, y_n) .$$

B. Der (globale) Fehler

Es lässt sich zeigen, dass für Differentialgleichungen (5.20), die eine Lipschitzbedingung $|f(t, y) - f(t, \tilde{y})| \leq L|y - \tilde{y}|$ für alle $t \in [a, b]$ und alle $y, \tilde{y} \in \mathbb{R}$ erfüllen, gilt:

$$(5.23) \quad |y(t_j) - y_j| \leq \frac{1}{2} h \max_{t \in [a, t_j]} |y''(t)| \frac{e^{L(t_j - a)} - 1}{L} .$$

Da sich der globale Fehler wie $O(h)$ verhält, sagt man, das Euler-Verfahren habe die **Fehlerordnung** (kurz **Ordnung**) 1. Damit ist im Prinzip die Anfangswertaufgabe vollständig gelöst. Zu einer vorgegebenen Toleranz TOL gibt es ein h , so dass für die mit dem Euler-Verfahren (5.22) mit diesem h berechneten y_j gilt

$$(5.24) \quad |y(t_j) - y_j| \leq TOL \text{ für } j = 0, 1, \dots, N .$$

In der Praxis wird man aber nicht mit der Formel (5.23) arbeiten. Meistens ist L nicht bekannt, und da die exakte Lösung $y(t)$ nicht bekannt ist, kann es schwierig sein, $y'(t)$ abzuschätzen. Selbst wenn y' existiert und abgeschätzt werden kann, und wenn L bekannt ist, ist die Abschätzung (5.23) meist zu schlecht, d.h. wählt man h^* so, dass gilt

$$\frac{1}{2} h^* \max_{t \in [a, b]} |y''(t)| \frac{e^{L(b-a)} - 1}{L} = TOL ,$$

so gilt wohl wegen (5.23) auch (5.24), aber es gibt ein $h_0 \gg h^*$, für das (5.24) auch noch gilt. Aus diesen Gründen sollte ein Programm versuchen, den Fehler zu schätzen statt ihn abzuschätzen. Leider ist dies auch nicht möglich, solange die Integration noch nicht bis b ausgeführt wird. Dies können wir uns wie folgt überlegen:

In der Abbildung 5.6 ist für drei Beispiele von Differentialgleichungen die Lösung des folgenden hypothetischen Verfahrens H eingezeichnet.

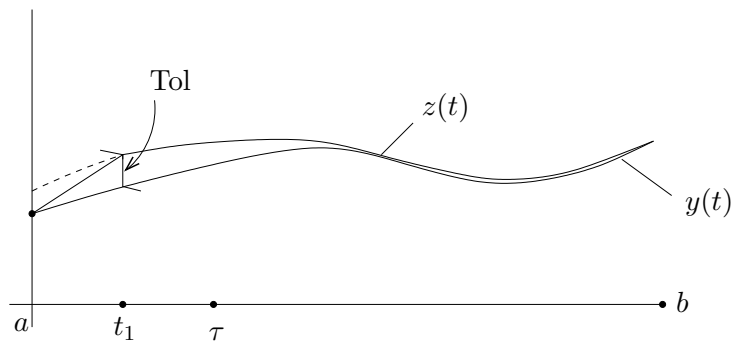


Fig. 5.6.a

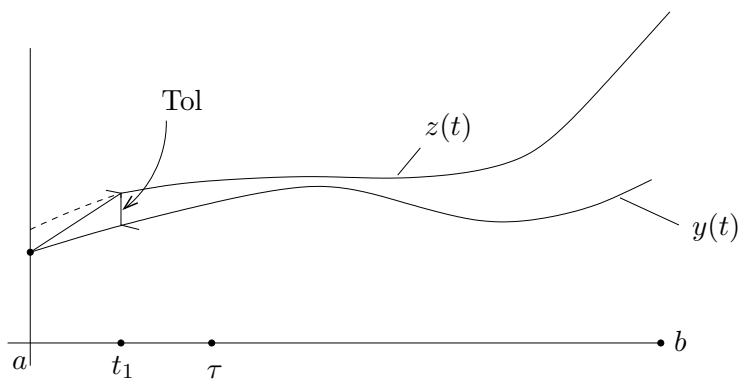


Fig. 5.6.b

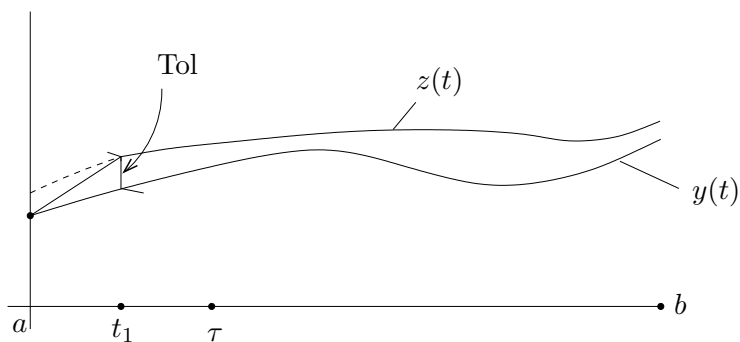


Fig. 5.6.c

Hypothetisches Verfahren H

H : Der erste Schritt ist ein Euler-Schritt. Danach ist das Verfahren exakt, d.h. $z(t)$ ist eine exakte Lösung der Differentialgleichung (5.20) durch die Anfangswerte $z(t_1) = y_1$.

In allen drei Fällen wurde genau derselbe Euler-Schritt ausgeführt. Die Beispiele sind so konstruiert, dass die Differentialgleichungen für $t < \tau$ identisch sind. Also kann das hypothetische Verfahren H nach einem Schritt **nicht** entscheiden, wie gross der Fehler bei $t = b$ sein wird (obwohl es für $t \geq t_1$ keine neuen Fehler mehr macht). In Fig. 5.6.b wurde der erste Schritt viel zu gross gemacht. In Fig. 5.6.a und 5.6.c ist wohl der Fehler bei $t = b$ innerhalb der Toleranz TOL , aber bei Fig. 5.6.a hätte der erste Schritt wesentlich grösser gemacht werden können, während er in Fig. 5.6.c zu gross ist, denn für gewisse $t \in (a, b)$ übersteigt der Fehler die vorgegebene Toleranz TOL . Aus diesen Gründen beschränken sich fast alle Programme darauf, den lokalen Fehler zu schätzen und diesen bezüglich einer vorgegebenen Toleranz klein zu halten.

C. Der lokale Fehler

Der lokale Fehler beim Ausführen des $n + 1$ -ten Schrittes ist in Fig. 5.2 eingezeichnet. Man beschreibt ihn wie folgt: Es sei $z(t)$ die exakte Lösung der Anfangswertaufgabe

$$(5.25) \quad z' = f(t, z), t \geq t_n,$$

$$(5.26) \quad z(t_n) = y_n .$$

Mit dem Euler-Verfahren berechnet man

$$(5.27) \quad y_{n+1} = y_n + hf(t_n, y_n) .$$

Die Differenz $z(t_n + h) - y_{n+1}$ heisst lokaler Fehler des Euler-Verfahrens beim $(n + 1)$ -ten Schritt. Man erkennt in Fig. 5.2, dass sich der globale Fehler aus dem lokalen Fehler und $z(t_n + h) - y(t_n + h)$ zusammensetzt.

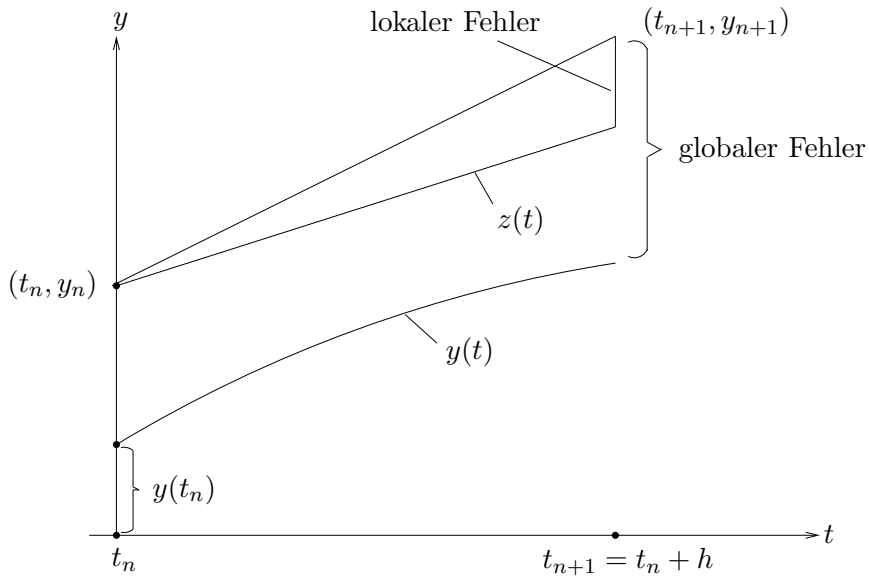


Fig. 5.7

Der zweite Beitrag ist unabhängig vom verwendeten Verfahren. Er hängt nur von der Differentialgleichung ab.

Wie gross ist der lokale Fehler?

Nach Taylor gilt (falls $z(t) \in C^3$)

$$(5.28) \quad z(t_n + h) = z(t_n) + hz'(t_n) + \frac{h^2}{2!}z''(t_n) + \frac{h^3}{3!}z'''(\xi)$$

für ein geeignetes $\xi \in [t_n, t_n + h]$.

Da $z(t)$ die Anfangswertaufgabe (5.25), (5.26) löst, gilt

$$(5.29) \quad \begin{aligned} z(t_n) &= y_n \\ z'(t_n) &= f(t_n, z(t_n)) = f(t_n, y_n) . \end{aligned}$$

Somit folgt aus (5.27), (5.28) und (5.29)

$$(5.30) \quad \begin{aligned} \text{lokaler Fehler} &= \varphi(t_n, h) := z(t_n + h) - y_{n+1} = \\ &= y_n + hf(t_n, y_n) + \frac{h^2}{2!}z''(t_n) + O(h^3) \\ &\quad - (y_n + hf(t_n, y_n)) \\ &= \frac{h^2}{2}z''(t_n) + O(h^3) . \end{aligned}$$

Bemerkung:

Man erkennt, dass sich der lokale Fehler wie $O(h^2)$ verhält, während der globale Fehler in (5.23) $O(h)$ war.

Ein Rechenprogramm benötigt eine zweite Formel, um den lokalen Fehler schätzen zu können. Es gibt viele solche Formeln. Wir greifen hier als Beispiel eine ganz einfache heraus. Es sei \hat{y}_{n+1} der Wert, der durch zweimaliges Anwenden der Euler-Formel mit Schritt $h/2$ entsteht. Also ist

$$Y = y_n + \frac{h}{2}f(t_n, y_n)$$

das Resultat nach einem Euler-Schritt mit $h/2$ und

$$(5.31) \quad \hat{y}_{n+1} = y_n + \underbrace{\frac{h}{2}f(t_n, y_n)}_Y + \frac{h}{2}f\left(t_n + \frac{h}{2}, \underbrace{y_n + \frac{h}{2}f(t_n, y_n)}_Y\right)$$

der Wert nach zwei Euler-Schritten.

Für den lokalen Fehler $\hat{\varphi}(t_n, h)$ findet man durch Taylor-Entwicklung nach h

$$\begin{aligned}
 \hat{\varphi}(t_n, h) &:= z(t_n + h) - \hat{y}_{n+1} = \\
 &= y_n + hf(t_n, y_n) + \frac{h^2}{2!} z''(t_n) + O(h^3) - \\
 &\quad - \left\{ y_n + hf(t_n, y_n) + \frac{h^2}{4} (f_t(t_n, y_n) + \right. \\
 &\quad \left. + f_y(t_n, y_n)f(t_n, y_n)) + O(h^3) \right\}, \\
 (5.32) \quad \hat{\varphi}(t_n, h) &= \frac{h^2}{4} z''(t_n) + O(h^3) .
 \end{aligned}$$

Hier wurde verwendet, dass man aus (5.25) durch Differenzieren

$$\begin{aligned}
 z''(t) &= f_t(t, z(t)) + f_y(t, z(t))z'(t) = \\
 &= f_t(t, z(t)) + f_y(t, z(t))f(t, z(t))
 \end{aligned}$$

erhält.

Aus (5.30) und (5.32) erhält man durch Subtraktion

$$(5.33) \quad \hat{\varphi}(h) = \hat{y}_{n+1} - y_{n+1} = \frac{h^2}{4} z''(t_n) + O(h^3)$$

eine Schätzung des lokalen Fehlers von \hat{y}_{n+1} und mit $2(\hat{y}_{n+1} - y_{n+1})$ eine Schätzung des lokalen Fehlers von y_{n+1} .

Bemerkung:

Wir können diese Schätzungen des lokalen Fehlers auch wie folgt verstehen: Aus

$$(5.30) \quad z(t_n + h) - y_{n+1} = \frac{h^2}{2} z''(t_n) + O(h^3)$$

und

$$(5.32) \quad z(t_n + h) - \hat{y}_{n+1} = \frac{h^2}{4} z''(t_n) + O(h^3)$$

lässt sich der $O(h^2)$ -Term eliminieren. Hierzu subtrahieren wir (5.30) vom Doppelten von (5.32) und erhalten

$$(5.34) \quad z(t_n + h) - \underbrace{(2\hat{y}_{n+1} - y_{n+1})}_{Y_{n+1}} = O(h^3) .$$

Somit haben wir eine neue Formel Y_{n+1} der Fehlerordnung 2 erhalten. Durch Einsetzen von (5.27) und (5.31) erhalten wir

$$(5.35) \quad Y_{n+1} := 2\hat{y}_{n+1} - y_{n+1} = y_n + hf\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}f(t_n, y_n)\right) .$$

Dies ist die sogenannte modifizierte Euler-Formel. Wenn wir mit ihr den lokalen Fehler schätzen, erhalten wir mit

$$Y_{n+1} - \hat{y}_{n+1} = \hat{y}_{n+1} - y_{n+1}$$

und

$$Y_{n+1} - y_{n+1} = 2(\hat{y}_{n+1} - y_{n+1})$$

gerade wieder die bereits hergeleiteten Schätzformeln für den lokalen Fehler von \hat{y}_{n+1} und y_{n+1} zurück.

D. Schrittweitensteuerung

An der in diesem Abschnitt hergeleiteten Euler-Formel und dem Schätzer $2(\hat{y}_{n+1} - y_{n+1})$ für den lokalen Fehler wollen wir das Prinzip der Schrittweitensteuerung studieren. Da pro Schritt zwei Auswertungen von f nötig sind, wird die Integration schneller und billiger, je grösser der Integrationsschritt h gewählt wird. Umgekehrt wird der Fehler grösser, je grösser h ist. Aus diesem Grund wird durch eine vorgegebene Toleranz TOL der Schritt h eingeschränkt. Es gibt somit eine optimale Wahl für die Schrittweite. Im allgemeinen ist eine konstante Schrittweite **nicht** optimal. Man wird stets variable Schritte $h_i = t_i - t_{i-1}$ zulassen.

Es sei nun eine Toleranz TOL vorgegeben. Wie wir schon gesehen haben können wir nicht erwarten, dass das numerische Resultat nach der gesamten Integration genauer als die vorgegebene Toleranz TOL ist. Vielmehr begnügen wir uns damit dass gilt:

$$(5.36) \quad \begin{aligned} |\text{lokaler Fehler pro Schritt}| &\lesssim TOL \\ \text{Schrittweitensteuerung:} &\quad \text{lokaler Fehler pro Schritt} \end{aligned}$$

Geht man davon aus dass sich die lokalen Fehler für kleine h ungefähr addieren, wählt man besser

$$(5.37) \quad \begin{aligned} |\text{lokaler Fehler pro Schritt}| \cdot \frac{1}{h} &\lesssim TOL \\ \text{Schrittweitensteuerung:} &\quad \text{lokaler Fehler pro Einheitsschritt} \end{aligned}$$

Im folgenden wollen wir (5.37) verwenden. Approximieren wir $\hat{\varphi}(h)$ durch

$$(5.38) \quad |\hat{\varphi}(h)| \approx \gamma h^2 ,$$

so lässt sich γ durch einen Probeschritt H schätzen. Es ist

$$(5.39) \quad \gamma \approx \frac{|\hat{\varphi}(H)|}{H^2} .$$

Nach (5.37) wäre die lokal optimale Wahl des Schrittes h

$$(5.40) \quad \frac{|\hat{\varphi}|}{h} = TOL .$$

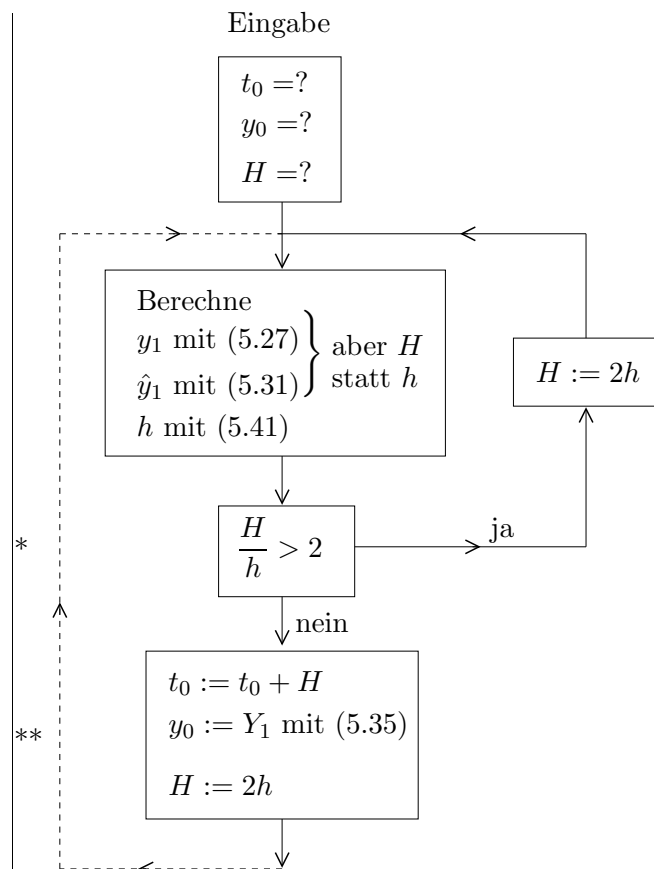
Einsetzen von (5.38) und (5.39) ergibt

$$\gamma h = \frac{|\hat{\varphi}(H)|}{H^2} \cdot h = TOL$$

und

$$(5.41) \quad h = \frac{TOL}{|\hat{\varphi}(H)|} \cdot H^2 .$$

Diese Formeln können wir nun für das folgende Programm verwenden, das wir durch ein Flussdiagramm darstellen.



Abfrage:

* Ist $\frac{H}{h} > 2$, so ist das optimale h (wesentlich) kleiner als das H , auf dem die Schätzung beruht. Es ist sogar zu befürchten, dass die asymptotischen Entwicklungen für dieses H nicht gelten. Deshalb wird der Schritt verworfen und als neuer Probeschritt $H := 2h$ verwendet. Ist $H/h \leq 2$, so gilt $H/2 \leq h$, und die asymptotischen Entwicklungen können akzeptiert werden. Ist $H/2 < h$, so darf der Schritt vergrößert werden. Da der Integrationschritt offenbar richtig war, wird er akzeptiert, und für den nächsten Schritt verwendet man die Schätzung für den optimalen Schritt. In der Praxis wird man auch einen Schritt akzeptieren, bei dem $H/2$ unwesentlich grösser als h ist. Aus diesem Grund werden wir im folgenden Beispiel den Schritt noch akzeptieren, wenn gilt $H/2 < 3h/2$, d.h. * wird ersetzt durch $H/h \geq 3$.

** Nach unserer Herleitung müsste hier $y_0 := \hat{y}_1$ stehen. Da man aber **ohne Mehrkosten** eine Formel zweiter Ordnung zur Verfügung hat, verwendet man diese. Damit ist unsere Fehlerschätzung meist zu pessimistisch.

BEISPIEL 5.4:

Wir lösen die Anfangswertaufgabe

$$y' = -200ty^2, t \in [-0.8, -0.2], \quad y(-0.8) = 1/65 .$$

Die exakte Lösung ist

$$y(t) = \frac{1}{1 + 100t^2} .$$

Wir verwenden das soeben beschriebene Rechenprogramm mit einer Toleranz $TOL = 6 \cdot 10^{-5}$, einem Startwert $h = 3 \cdot 10^{-3}$ auf einem IMS 8000-Rechner mit einfacher Genauigkeit. Der Fehler bei $t = -0.2$ ist $6 \cdot 10^{-6}$. Es werden insgesamt 13006 Funktionsauswertungen verwendet, und die kleinste Integrationschrittweite ist $0.136 \cdot 10^{-4}$. Als Vergleich wurde das Problem mit konstanter Schrittweite und Formel (5.35) berechnet.

Schrittweite	Funktionsauswertungen	Fehler bei $t = -0.2$
$0.6 \cdot 2^{-5}$	64	-0.005427
$0.6 \cdot 2^{-6}$	128	-0.001487
$0.6 \cdot 2^{-7}$	256	-0.000382
$0.6 \cdot 2^{-8}$	512	-0.000112
$0.6 \cdot 2^{-9}$	1024	-0.000053
$0.6 \cdot 2^{-10}$	2048	0.000049
$0.6 \cdot 2^{-11}$	4096	0.000096
$9.22651 \cdot 10^{-5}$	130096	-0.000082

Aus der letzten Zeile dieser Tabelle erkennt man, dass bei konstantem Schritt bei gleichem Rechenaufwand das Programm mit variabler Schrittweite um einen Faktor 10 genauer ist. Man erkennt, dass der absolute Fehler bei kleiner werdenden Schritt abfällt bis zu einem optimalen Schritt ($H_{\text{optimal}} \sim 5.859 \cdot 10^{-4}$), und danach steigt er wieder an. Extrapolation der obigen Tabellenwerte bringt keine Verbesserung der Genauigkeit, so dass die Genauigkeit des Programms mit Schrittweitensteuerung mit Integration mit konstanter Schrittweite nicht erreicht wird.

Bemerkung:

Es lässt sich zeigen, dass es sich bei kleinen Toleranzen TOL lohnt Verfahren hoher Ordnung zu verwenden. In den beiden nächsten Abschnitten geben wir nur noch Formeln mit höherer Fehlerordnung an. Die ausgereiften Programme haben Familien von Formeln und Schätzer des lokalen Fehlers zur Verfügung und wählen jeweils die lokal günstigste Formel.

5.3 Einschritt-Formeln

A. Der lokale Fehler

Unter Einschritt-Formel verstehen wir eine Vorschrift, bei der aus y_n, t_n und h nur unter Zuhilfenahme von $f(t, y)$ ein neuer Punkt $y_{n+1}, t_{n+1} = t_n + h$ berechnet wird. Die Formel wird meistens in der Form

$$(5.42) \quad y_{n+1} = y_n + h\phi(t_n, y_n, h)$$

geschrieben. ϕ ist hier eine Vorschrift, die von f abhängen kann. Als einfachstes Beispiel denke man an das Euler-Verfahren

$$y_{n+1} = y_n + hf(t_n, y_n) .$$

Definition 5.4. Ist $z(t)$ die exakte Lösung des Anfangswertproblems

$$(5.43) \quad \begin{cases} z'(t) &= f(t, z(t)) \\ z(t_n) &= y_n \end{cases} \quad t \geq t_n ,$$

dann heisst $z(t_{n+1}) - y_{n+1}$ der **lokale Fehler**. Man sagt, die Formel (5.42) habe die **Fehlerordnung p** (kurz **Ordnung p**) falls gilt

$$(5.44) \quad z(t_n + h) - y_{n+1} = O(h^{p+1}) .$$

Wir werden hier nur Formeln (5.42) betrachten, bei denen man aus (5.44) zeigen kann, dass für den globalen Fehler gilt

$$(5.45) \quad |y(t_n) - y_n| = O(h^p) .$$

Sind die Lösungen der Differentialgleichungen glatt, so erweisen sich Rechenprogramme, die Formeln hoher Ordnung verwenden, als günstiger. Das gleiche gilt für kleine Toleranzen. Wir geben deshalb hier verschiedene Formeln höherer Fehlerordnung an.

B. Potenzreihen-Verfahren

Eine einfache Möglichkeit, Formeln hoher Ordnung p zu konstruieren, ist die folgende: Ist $z(t) \in C^{p+1}$, so gilt nach dem Satz von Taylor

$$(5.46) \quad z(t_n + h) = y_n + \sum_{j=1}^p \frac{h^j}{j!} z^{(j)}(t_n) + \frac{h^{p+1}}{(p+1)!} z^{(p+1)}(\xi)$$

für ein geeignetes $\xi \in [t_n, t_n + h]$. Mit

$$(5.47) \quad \phi(t_n, y_n, h) := \sum_{j=1}^p \frac{h^{j-1}}{j!} z^{(j)}(t_n)$$

erhalten wir die Formel

$$(5.48) \quad y_{n+1} = y_n + h\phi(t_n, y_n, h) = y_n + \sum_{j=1}^p \frac{h^j}{j!} z^{(j)}(t_n) .$$

Diese Formel hat offensichtlich die Ordnung p .

Wie berechnet man $z^{(j)}(t_n)$?

Da $z(t_n) = y_n$ und $z(t)$ die Differentialgleichung

$$(5.49) \quad z'(t) = f(t, z(t))$$

erfüllt, erhält man sofort

$$(5.50) \quad z'(t_n) = f(t, y_n) .$$

Differenzieren von (5.49) ergibt

$$(5.51) \quad z''(t) = \frac{\partial}{\partial t} f(t, z(t)) + \frac{\partial}{\partial z} f(t, z(t)) \cdot z'(t) .$$

Setzt man in (5.51) $t = t_n$, so ist die rechte Seite von (5.51) bekannt und man erhält $z''(t_n)$. Nochmaliges Differenzieren ergibt

$$(5.52) \quad \begin{aligned} z'''(t) = & \frac{\partial^2}{\partial t^2} f(t, z(t)) + 2 \frac{\partial^2}{\partial t \partial z} f(t, z(t)) z'(t) + \\ & + \frac{\partial^2}{\partial z^2} f(t, z(t)) (z'(t))^2 + \frac{\partial}{\partial z} f(t, z(t)) z''(t) . \end{aligned}$$

Wiederum ist die rechte Seite in (5.52) für $t = t_n$ bekannt, und man kann so $z'''(t_n)$ berechnen. Weiteres Differenzieren von (5.52) liefert die gewünschten Ableitungen. Das Verfahren kann auch wie folgt beschrieben werden: Mit der Abkürzung

$$a_j = \frac{z^{(j)}(t_n)}{j!}$$

macht man für die exakte Lösung $z(t)$ den **Potenzreihen-Ansatz**

$$(5.53) \quad z(t) = \sum_{j=0}^{\infty} a_j (t - t_n)^j .$$

Einsetzen in die Differentialgleichung (5.43) ergibt

$$(5.54) \quad \sum_{j=0}^{\infty} a_{j+1} (j+1) (t - t_n)^j = f(t_n + (t - t_n), \sum_{j=0}^{\infty} a_j (t - t_n)^j) .$$

Entwickelt man die rechte Seite von (5.54) nach Potenzen von $(t - t_n)$, so erhält man durch Koeffizientenvergleich aus (5.54) Rekursionsformeln für a_j . Da $a_0 = y_n$ bekannt ist, kann man diese Formel rekursiv lösen. Besteht $f(t, y)$ aus den Grundrechenoperationen $+$, $-$, \cdot , \div und elementaren Funktionen wie $\sin(\cdot)$, $\cos(\cdot)$, $\exp(\cdot)$, $\sqrt{\cdot}$, \dots , so können die a_j sehr schnell rekursiv berechnet werden, siehe z.B. G. Wanner [1969].

Man hat hier eine ganze Familie von Formeln zur Verfügung. Zu jeder natürlichen Zahl s gibt es eine Formel mit der Ordnung $p = s$. Der lokale Fehler lässt sich wie folgt schätzen:

$$(5.55) \quad \begin{aligned} \text{lokaler Fehler} &= \frac{h^{p+1}}{(p+1)!} z^{(p+1)}(\zeta) \\ &\approx h^{p+1} \frac{z^{(p+1)}(t_n)}{(p+1)!} = h^{p+1} a_{p+1} , \end{aligned}$$

d.h. man muss in der Rekursion für a_j nur noch ein weiteres a_j ausrechnen, um eine Schätzung für den lokalen Fehler zu erhalten. Mit diesen Formeln lassen sich Programme schreiben, die die Formeln und die Schrittweite so wählen, dass die Integration möglichst schnell ausgeführt wird. Solche Programme sind für Probleme der Himmelsmechanik (Satellitenbahnen, Raumsonden, Planetenbahnen) sehr geeignet.

C. Runge-Kutta-Verfahren

In B haben wir Verfahren beliebig hoher Ordnung berechnet. Schon an der Formel (5.51) für z'' erkennt man, dass die partiellen Ableitungen von f benötigt werden. Hier wollen wir versuchen, Formeln der Ordnung p grösser als 1 zu konstruieren, ohne partielle Ableitungen zu verwenden. Eine solche Formel haben wir bereits in Abschnitt 5.2 C (5.35) gefunden:

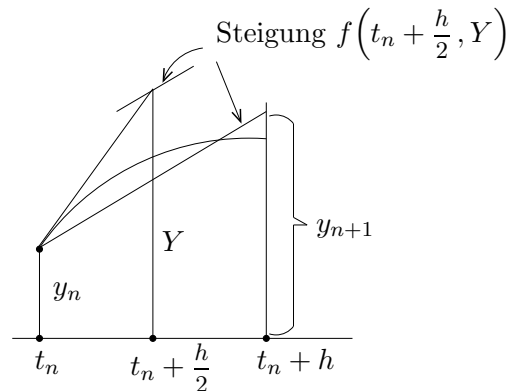


Fig. 5.8

Modifizierter Euler

$$(5.56) \quad Y = y_n + \frac{h}{2} f(t_n, y_n)$$

$$(5.57) \quad y_{n+1} = y_n + hf\left(t_n + \frac{h}{2}, Y\right)$$

In Paragraph 5.2 C wurde bereits gezeigt, dass dieses Verfahren die Ordnung 2 hat. Für die triviale Differentialgleichung mit $f(t, y) = f(t)$ ist dies gerade die Mittelpunktsregel, die Ordnung 2 hat.

Statt mit der Steigung des Richtungsfeldes in der Mitte des Integrationsintervalles einen Schritt auszuführen, wählen wir das arithmetische Mittel der Steigungen an den Endpunkten. Dies ergibt die

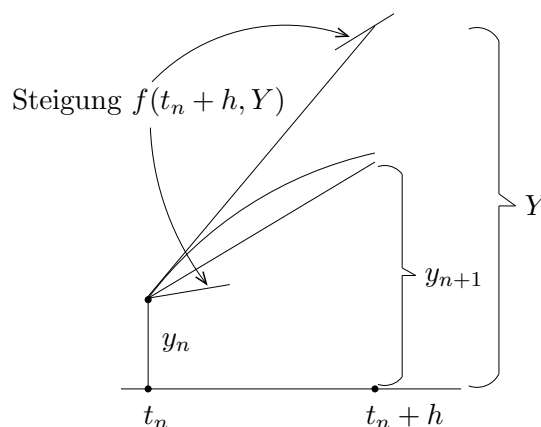
Formel von Heun

Fig. 5.9

$$(5.58) \quad Y = y_n + hf(t_n, y_n) \quad (\text{Euler - Schritt})$$

$$(5.59) \quad y_{n+1} = y_n + \frac{h}{2}(f(t_n, y_n) + f(t_n + h, Y))$$

Die Ordnung des Verfahrens ist 2. Für die triviale Differentialgleichung mit $f(t, y) = f(t)$ ist dies gerade die Trapezregel.

Man kann diese Idee weiterführen, indem man noch mehr Zwischenpunkte ausrechnet und eine Mittelung der Steigungen vornimmt. Dies führt zum Beispiel auf das

Klassische 4-stufige Runge-Kutta-Verfahren

$$(5.60) \quad \begin{cases} Y_1 = y_n \\ Y_2 = y_n + \frac{h}{2} f(t_n, Y_1) \\ Y_3 = y_n + \frac{h}{2} f(t_n + \frac{h}{2}, Y_2) \\ Y_4 = y_n + hf(t_n + \frac{h}{2}, Y_3) \end{cases}$$

$$(5.61) \quad \begin{aligned} y_{n+1} = & y_n + h \left(\frac{1}{6} f(t_n, Y_1) + \frac{1}{3} f(t_n + \frac{h}{2}, Y_2) + \right. \\ & \left. + \frac{1}{3} f(t_n + \frac{h}{2}, Y_3) + \frac{1}{6} f(t_n + h, Y_4) \right) . \end{aligned}$$

Die Fehlerordnung ist 4. Für die Differentialgleichung mit $f(t, y) = f(t)$ reduziert sich (5.61) zur sogenannten Simpson-Regel zum Berechnen von Integralen.

Die drei angegebenen Formeln sind alle von der folgenden Struktur:

Allgemeines s -stufiges Runge-Kutta-Verfahren

$$(5.62) \quad Y_i = y_n + h \sum_{j=1}^{i-1} a_{ij} f(t_n + c_j h, Y_j), \quad i = 1, 2, \dots, s,$$

$$(5.63) \quad y_{n+1} = y_n + h \sum_{j=1}^s b_j f(t_n + c_j h, Y_j) .$$

Hier sind a_{ij} , c_j und b_j konstante reelle Zahlen, die das Verfahren vollständig beschreiben. Man ordnet sie meistens wie folgt an:

$$\begin{array}{c|cccc}
 c_1 & & & & \\
 c_2 & a_{21} & & & \\
 c_3 & a_{31} & a_{32} & & \\
 \vdots & & & & \\
 c_i & a_{i1} & a_{i2} & \dots & a_{i,i-1} \\
 \vdots & & & & \\
 c_s & a_{s1} & a_{s2} & \dots & a_{s,s-1} \\
 \hline
 & b_1 & b_2 & \dots & b_{s-1} & b_s
 \end{array}$$

Es ist sinnvoll, stets zu verlangen, das gilt

$$c_i = \sum_{j=1}^{i-1} a_{ij}$$

Da die Formeln für die spezielle Differentialgleichung $f(t) \equiv 1$ exakt sein muss, gilt stets

$$\sum_{i=1}^s b_i = 1$$

Mit dieser Notation schreiben sich die bereits behandelten Formeln wie folgt:

Modifizierter Euler $s = 2, p = 2$

$$\begin{array}{c|cc}
 0 & & \\
 1/2 & 1/2 & \\
 \hline
 & 0 & 1
 \end{array}$$

Heun $s = 2, p = 2$

$$\begin{array}{c|cc}
 0 & & \\
 1 & 1 & \\
 \hline
 & 1/2 & 1/2
 \end{array}$$

Klassischer 4-stufiger Runge-Kutta $s = 4, p = 4$

$$\begin{array}{c|cccc}
 0 & & & & \\
 1/2 & 1/2 & & & \\
 1/2 & 0 & 1/2 & & \\
 1 & 0 & 0 & 1 & \\
 \hline
 & 1/6 & 1/3 & 1/3 & 1/6
 \end{array}$$

Wie im Abschnitt 5.2 demonstriert wurde, ist es im allgemeinen günstiger, eine Schrittweitensteuerung in ein Rechenprogramm einzubauen. Man benötigt somit zu jeder Formel eine Fehlerschätzungsformel. Dies liesse sich im Falle des klassischen Runge-Kutta-Verfahrens analog zu unserem Vorgehen in Abschnitt 5.2 lösen. Man macht zwei Runge-Kutta-Schritte der Schrittweite $h/2$. Hierzu würde man aber **zusätzlich** 7 Funktionsauswertungen benötigen. Formel und Fehlerschätzung würden pro Schritt 11 Funktionsauswertungen benötigen. Mit sogenannten **eingebetteten** Runge-Kutta-Formeln kann man mit wesentlich weniger Funktionsauswertungen auskommen. Die Idee ist die, dass man die

Stufenzahl erhöht und die Koeffizienten so wählt, dass zwei Formeln mit hoher Ordnung entstehen. Die eine wird zur Integration verwendet, die andere zur Schätzung des Fehlers. Wir geben hier die Formel von Dormand und Price an.

Formeln von Dormand und Price

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	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
y_1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
\hat{y}_1	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Bemerkung:

Die in Abschnitt 5.2 hergeleiteten Formeln können auch als eingebettete Runge-Kutta-Formeln gesehen werden. Die Formeln (5.31) und (5.35) in 5.2 C haben in der Runge-Kutta-Notation die Form

0			
1/2	1/2		
1/2	1/2	Formel für zwei Euler-Schritte mit $h/2$ (Ordnung 1)	
0	1	modifizierter Euler (Ordnung 2)	

Die Formeln von Dormand und Price können genauso für ein Programm mit Schrittweitensteuerung verwendet werden, wie dies mit obigem Beispiel in Abschnitt 5.2 gemacht wurde.

D. Extrapolationsverfahren

In Paragraph 5.2 haben wir am einfachen Beispiel des Euler-Verfahrens einen Extrapolationsschritt ausgeführt (Formel (5.35)). Dies liesse sich natürlich weiterführen, indem man nochmals mit einem Schritt kleiner $h/2$ mit Euler integriert und dann extrapoliert. Dies lohnt sich nicht, denn in der Fehlerentwicklung treten alle Potenzen auf. Nun hat aber Gragg gezeigt, dass es Formeln gibt, deren Fehlerentwicklung nur gerade Potenzen in h zulassen. In diesem Falle lohnt sich die Extrapolation. Man erhält das Verfahren von Gragg-Bulirsch, J. Stoer, R. Bulirsch [1973]. Wir verzichten wir an dieser Stelle auf eine genauere Darstellung.

5.4 Formeln für Mehrschrittverfahren

A. Konzept

Die bis jetzt behandelten Einschrittverfahren verwenden nur y_n und $f(t_n, y_n)$, um y_{n+1} zu berechnen. Um eine Formel der Ordnung p zu erhalten, benötigt man mindestens p Funktionsauswertungen, häufig sogar mehr. Die Frage stellt sich, ob durch Verwenden alter Information (d.h. in Punkten mit $t < t_n$) Formeln hoher Ordnung erreicht werden können mit weniger Funktionsauswertungen als bei Einschrittverfahren. Der Einfachheit halber wollen wir im folgenden voraussetzen, dass die Schrittweite konstant ist, d.h.

$$h = (b - a)/N, t_i = a + ih, i = 0, 1, \dots, N.$$

Ein sogenanntes k -Schrittverfahren verwendet nun nicht nur $y_n, f(t_n, y_n)$, um y_{n+1} zu berechnen, sondern die Werte

$$y_{n-i}, f(t_{n-i}, y_{n-i}), i = 0, 1, \dots, k-1.$$

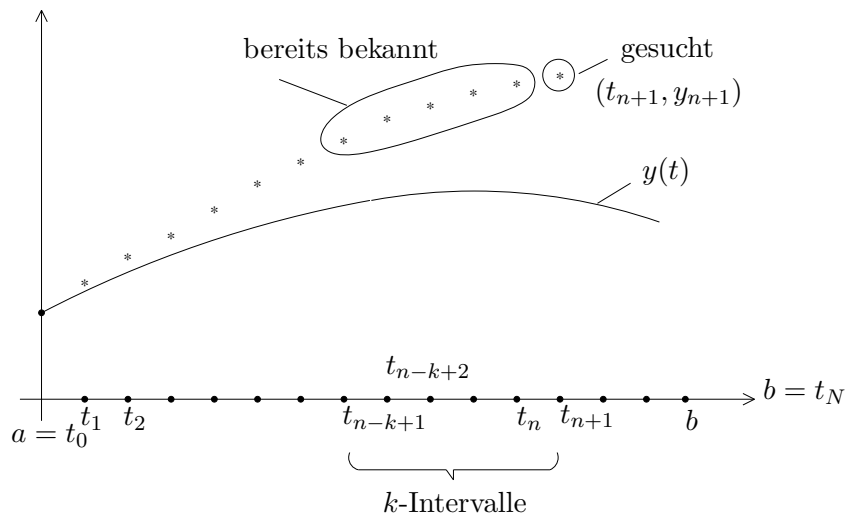


Fig. 5.10

Da in der Formel Information der Lösung über k Teilintervalle der Länge h verwendet wird, nennt man die Verfahren k -Schrittverfahren. Naturgemäß werden sich bei der Implementierung lösbar Schwierigkeiten beim Starten und bei der Änderung der Schrittweite ergeben, siehe z.B. C.W. Gear [1971], L.F. Shampine, M.K. Gordon [1974]. In den nächsten beiden Abschnitten geben wir zwei Formeltypen an, die Grundlage der besten Rechnerprogramme sind.

B. Adams-Verfahren

Für die exakte Lösung $y(t)$ der Differentialgleichung gilt

$$(5.64) \quad y'(t) = f(t, y(t)) \quad .$$

Integration dieser Gleichung ergibt

$$(5.65) \quad \int_{t_n}^{t_n+h} y'(t) dt = y(t_n+h) - y(t_n) = \int_{t_n}^{t_n+h} f(t, y(t)) dt \quad .$$

Da die Funktion $f(t, y(t))$ nicht bekannt ist, ersetzen wir sie durch ein Interpolationspolynom $P(t)$ und integrieren dieses exakt. Man erhält auf diese Weise die Formeln

$$(5.66) \quad y_{n+1} - y_n = \int_{t_n}^{t_n+h} P(t) dt \quad .$$

Abkürzend führen wir die folgende Notation ein:

$$f_m := f(t_m, y_m)$$

BEISPIEL 5.5:

Ersetze $f(t, y(t))$ durch ein konstantes Polynom, d.h.

$$(5.67) \quad P(t) = f_n \quad .$$

Dies ergibt

$$(5.68) \quad y_{n+1} - y_n = \int_{t_n}^{t_n+h} f_n dt = h f_n \quad ,$$

also das **Euler-Verfahren**. Interpoliert man nicht bei t_n, f_n , sondern an der noch unbekannt Stelle t_{n+1}, f_{n+1} , d.h.

$$(5.69) \quad P(t) = f_{n+1} \quad ,$$

so erhält man

$$(5.70) \quad y_{n+1} - y_n = h f_{n+1} = h f(t_{n+1}, y_{n+1}) \quad .$$

Dies ist das **implizite Euler-Verfahren**. □

BEISPIEL 5.6:

Lineare Interpolation durch die Punkte (t_{n-1}, f_{n-1}) und (t_n, f_n) ergibt

$$(5.71) \quad P(t) = f_n \frac{t - t_{n-1}}{h} + f_{n-1} \frac{t - t_n}{-h} .$$

Damit ergibt sich

$$(5.72) \quad \begin{aligned} y_{n+1} - y_n &= \int_{t_n}^{t_n+h} P(t) dt = \\ &= \frac{(t - t_{n-1})^2}{2h} \Big|_{t_n}^{t_n+h} f_n + \frac{(t - t_n)^2}{-2h} f_{n-1} \Big|_{t_n}^{t_n+h} = \\ &= h \left(\frac{3}{2} f_n - \frac{1}{2} f_{n-1} \right) . \end{aligned}$$

Dies ist die sogenannte Adams-Bashforth-Zweischrittformel. \square

Interpoliert man (t_n, f_n) und den noch unbekanntem Punkt (t_{n+1}, f_{n+1}) , so liefert dies

$$\begin{aligned} P(t) &= f_{n+1} \frac{t - t_n}{h} + f_n \frac{t - t_{n+1}}{-h} \\ y_{n+1} - y_n &= \int_{t_n}^{t_n+h} P(t) dt = \\ &= \frac{(t - t_n)^2}{2h} \Big|_{t_n}^{t_n+h} f_{n+1} + \frac{(t - t_{n+1})^2}{-2h} \Big|_{t_n}^{t_n+h} f_n = \\ &= h \left(\frac{1}{2} f_{n+1} + \frac{1}{2} f_n \right) . \end{aligned}$$

In der ausgeschriebenen Form

$$(5.73) \quad y_{n+1} - y_n = h \left(\frac{1}{2} f(t_{n+1}, y_{n+1}) + \frac{1}{2} f(t_n, y_n) \right)$$

erkennt man, dass dies eine implizite Gleichung für den zu bestimmenden Wert y_{n+1} ist. Die Formel heisst **Trapezregel**.

Adams-Bashforth Formeln

Ersetzt man $f(t, y(t))$ durch das Interpolationspolynom $P(t)$ vom Grad q durch die Punkte

$$(t_{n-i}, f_{n-i}), i = 0, 1, \dots, q,$$

so findet man wie im obigen Beispiel die **Adams-Bashforth-Formeln**

$$(5.74) \quad y_{n+1} = y_n + \sum_{i=0}^q \beta_{qi} f_{n-i} .$$

Tabelle für die Koeffizienten β_{qi}

	i=0	1	2	3	
β_{0i}	1				Euler
$2\beta_{1i}$	3	-1			Formel (5.72)
$12\beta_{2i}$	23	-16	5		
$24\beta_{3i}$	55	-59	37	-9	

Die Ordnung der entstehenden Formel ist $q + 1$.

Adams-Moulton-Formeln

Ersetzt man $f(t, y(t))$ durch das Interpolationspolynom $P(t)$ vom Grad q durch die Punkte

$$(t_{n+1-i}, f_{n+1-i}), i = 0, 1, \dots, q,$$

so findet man die Adams-Moulton-Formeln

$$(5.75) \quad y_{n+1} - y_n = h \sum_{i=0}^q \beta_{qi}^* f_{n+1-i},$$

die die Fehlerordnung $q + 1$ haben.

Tabelle für die Koeffizienten β_{qi}

	i=0	1	2	3	
β_{0i}^*	1				impliziter Euler
$2\beta_{1i}^*$	1	1			Trapezregel
$12\beta_{2i}^*$	5	8	-1		
$24\beta_{3i}^*$	9	19	-5	1	

Bemerkung:

Statt in (5.65) über das Intervall $[t_n, t_{n+1}]$ zu integrieren, hätte man auch über das Intervall $[t_{n-1}, t_{n+1}]$ integrieren können. Die meisten der so entstehenden Formeln weisen gewisse Stabilitätsprobleme auf. Die wichtigste Formel dieser ganzen Klasse von Formeln ist die

Mittelpunktsregel

$$(5.76) \quad y_{n+1} - y_{n-1} = 2hf(t_n, y_n) .$$

Sie hat die Ordnung 2. Durch geschicktes Verwenden dieser Formel kann erreicht werden, dass die Fehlerentwicklung eine gerade Funktion in h ist. Deshalb wird diese Formel als Grundlage für das in Abschnitt 5.2 D angedeutete Extrapolationsverfahren verwendet.

C. Rückwärtsdifferentiations-Formeln

Diese Formeln heissen auch BDF (**B**ackward **D**ifferentiation **F**ormulas) oder Gear-Formeln.

Idee: Interpoliere $(t_{n+1}, y_{n+1}), (t_n, y_n), \dots, (t_{n-q}, y_{n-q})$ durch ein Polynom $P(t)$ vom Grad $q + 1$. Setze

$$(5.77) \quad P'(t_{n+1}) = f(t_{n+1}, y_{n+1}) .$$

BEISPIEL 5.7: $q = 0$

Hier ist

$$P(t) = y_{n+1} + \frac{y_{n+1} - y_n}{h}(t - t_{n+1})$$

und somit

$$P'(t_{n+1}) = \frac{y_{n+1} - y_n}{h} = f(t_{n+1}, y_{n+1})$$

Also ist

$$(5.78) \quad y_{n+1} - y_n = hf(t_{n+1}, y_{n+1})$$

eine Formel zur Berechnung von y_{n+1} . Dies ist gerade die implizite Euler-Formel.

$q = 1$:

Mit der Newtonschen Interpolationsformel erhält man

$$P(t) = y_{n+1} + (t - t_{n+1})\frac{y_{n+1} - y_n}{h} + (t - t_{n+1})(t - t_n)\frac{y_{n+1} - 2y_n + y_{n-1}}{2h^2} .$$

Also ist

$$P'(t_{n+1}) = \frac{y_{n+1} - y_n}{h} + \frac{y_{n+1} - 2y_n + y_{n-1}}{2h} = f(t_{n+1}, y_{n+1}) .$$

Die Formel für die Berechnung von y_{n+1} lautet

$$(5.79) \quad y_{n+1} - \frac{4}{3}y_n + \frac{1}{3}y_{n-1} = \frac{2}{3}hf(t_{n+1}, y_{n+1}) .$$

Allgemein besitzen die so hergeleiteten Formeln die Form

$$(5.80) \quad \sum_{i=0}^{q+1} \alpha_{qi} y_{n+1-i} = h\beta_q f(t_{n+1}, y_{n+1}) .$$

	i=0	1	2	3	4	5	6	β_q	$p = q + 1$
α_{1i}	1	-1						1	1
$3\alpha_{2i}$	3	-4	1					2/3	2
$11\alpha_{3i}$	11	-18	9	-2				6/11	3
$25\alpha_{4i}$	25	-48	36	-16	3			12/25	4
$137\alpha_{5i}$	137	-300	300	-200	75	-12		60/137	5
$147\alpha_{6i}$	147	-360	450	-400	225	-72	10	60/147	6

Bemerkung:

Die Formeln (5.80) sollen nie für $q \geq 6$ angewendet werden, da sie in diesem Fall **nicht** ein konvergentes Verfahren liefern. Damit ist die Fehlerordnung der brauchbaren Formeln (5.80) auf 6 beschränkt.

D. Bemerkungen zur Implementation

Alle in B und C hergeleiteten Formeln haben die Form

$$(5.81) \quad \begin{aligned} \alpha_0 y_n + \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + \cdots + \alpha_k y_{n-k} &= \\ &= h(\beta_0 f_n + \beta_1 f_{n-1} + \cdots + \beta_k f_{n-k}), \end{aligned}$$

wobei α_i, β_i konstante reelle Zahlen sind und

$$t_j = a + jh, f_j := f(t_j, y_j)$$

gilt. Wegen der besseren Übersichtlichkeit in den Indizes haben wir diese gegenüber Abschnitt A, B und C um 1 verschoben. Notwendig für die Brauchbarkeit der Formeln (5.81) ist, dass für das Polynom

$$(5.82) \quad \rho(\zeta) = \sum_{i=0}^k \alpha_i \zeta^{k-i}$$

die folgende Stabilitätsbedingung erfüllt ist:

Stabilitätsbedingung:

Der Betrag der Nullstellen von $\rho(\zeta)$ übersteigt 1 nicht. Ist er gleich 1, so ist die Nullstelle einfach.

Man nennt Formeln, die die Stabilitätsbedingung erfüllen, **stabil**. Die Adams-Verfahren mit $\rho(\zeta) = \zeta^k - \zeta^{k-1} = (\zeta - 1)\zeta^{k-1}$ sind stabil, während die Rückwärtsdifferenzierungsformeln der Ordnung grösser 6 nicht stabil sind. Eine Formel (5.81) hat die Ordnung p , falls für jede Funktion $z(t) \in C^{p+1}$ gilt

$$(5.83) \quad \sum_{j=0}^k \alpha_j z(t - jh) - h \sum_{j=0}^k \beta_j z'(t - jh) = ch^{p+1} z^{(p+1)}(t) + O(h^{p+1}) .$$

Ist die Formel stabil so ist der globale Fehler von der Ordnung $O(h^p)$.

Wie wird die Formel (5.81) verwendet?

Man nimmt an, $(y_{n-1}, f_{n-1}), (y_{n-2}, f_{n-2}), \dots, (y_{n-k}, f_{n-k})$ seien bereits berechnet. Man verwendet (5.81) nun, um y_n zu berechnen. Man unterscheidet zwei Typen von Formeln:

Explizite Formeln, Prädiktoren

Ist $\beta_0 = 0$, so lässt sich (5.81) explizit nach y_n auflösen:

$$(5.84) \quad y_n = \left(- \sum_{i=1}^k \alpha_i y_{n-i} + h \sum_{i=1}^k \beta_i f_{n-i} \right) / \alpha_0 .$$

Beispiele sind die Adams-Bashforth-Formeln.

Implizite Formeln, Korrektoren

Ist $\beta_0 \neq 0$, so ist (5.81) eine im allgemeinen nichtlineare Gleichung für y_n :

$$(5.85) \quad y_n - h \frac{\beta_0}{\alpha_0} f(t_n, y_n) = d ,$$

wobei

$$d := \left(- \sum_{i=1}^k \alpha_i y_{n-i} + h \sum_{i=1}^k \beta_i f_{n-i} \right) / \alpha_0$$

unabhängig vom gesuchten y_n ist.

Beispiele sind die Adams-Moulton-Formeln sowie die Rückwärtsdifferenzierungsformeln. Ist die Differentialgleichung harmlos (nicht steif), so wird (5.85) mit einer Fixpunktiteration der Form

$$y_n^{[\nu+1]} = h \frac{\beta_0}{\alpha_0} f(t_n, y_n^{[\nu]}) + d$$

gelöst, wobei man sich den Startwert $y_n^{[0]}$ über einen Prädiktor beschafft. Häufig genügen ein bis zwei Iterationen.

Das Verfahren von Heun (Paragraph 5.3 C, (5.58),(5.59)) kann man auffassen als einen Iterationsschritt mit der Trapezregel, wobei der Startwert mit Euler bestimmt wurde. Ist die Differentialgleichung steif (siehe Abschnitt 5.5), so verwendet man die Rückwärtsdifferenzierungsformeln und löst (5.85) mit dem Newton-Verfahren.

Die Formeln (5.81) benötigen (y_{n-j}, f_{n-j}) für $j = 1, 2, \dots, k$. Diese sind beim Start der Integration natürlich nicht vorhanden. Die heute bekannten effizienten Programme besitzen ganze Familien von Formeln verschiedener Fehlerordnung, zum Beispiel die **Adams-Formeln** für nicht steife Differentialgleichungen und die **Rückwärtsdifferenzierungsformeln (BDF)** für steife Differentialgleichungen.

Beim Start werden zunächst Formeln mit kleinem k , also $1, 2, \dots$, verwendet, und erst später werden grosse k (zum Teil bis $k \approx 15$) benutzt. Bei der Implementierung dieser Formeln ist die Schreibweise (5.81) respektive (5.84) und (5.85) aus verschiedenen Gründen ungünstig (Schrittänderung, simultane Berechnung mehrerer Formeln). Man verwendet die sogenannte Nordsieck-Darstellung. Für genauere Hinweise über die Implementierung siehe C.W. Gear [1971]. Obwohl man zunächst denken würde, eine Runge-Kutta Formel sei wesentlich einfacher zu programmieren als ein Mehrschrittverfahren mit variabler Schrittweite, stellt sich dann aber heraus, dass die ausgereiften Programme wegen der diffizilen Schrittweitenwahl alle recht umfangreich werden. Aus diesem Grunde sollte man sich zunächst stets auf bekannte Programme stützen. Im nächsten Abschnitt beschreiben wir kurz verschiedene bekannte Programme.

5.5 Steife Differentialgleichungen

Treten keine Rundungsfehler auf, so kann für h genügend klein der Integrationsfehler beliebig klein gemacht werden (siehe z.B. (5.23) in Abschnitt 5.2 B). Selbst wenn man, wie das in 5.2 beschrieben wurde, den Schritt h variabel wählt, um die Anzahl der Funktionsauswertungen niedrig zu halten, kann folgendes geschehen: Völlig unwesentliche Beiträge in der exakten Lösung der Differentialgleichung zwingen ein numerisches Verfahren einen sehr kleinen Integrationsschritt zu wählen. Wir beschreiben dieses Phänomen am Beispiel 3 der erzwungenen Schwingung eines stark gedämpften Schwingers. Löst man das Anfangswertproblem

$$(5.86) \quad \begin{aligned} \frac{dy_1}{dt} &= y_2 =: f_1(t, y_1, y_2) \\ \frac{dy_2}{dt} &= -156.25y_1 - 200y_2 + 80 \cos t + 156.25 =: f_2(t, y_1, y_2), \end{aligned}$$

$$(5.87) \quad y_1(0) = 5, y_2(0) = -100$$

für $t \in [0, 5]$ mit einem Programm für nicht steife Differentialgleichungen, so ist dies dank der Schrittweitensteuerung möglich, aber die Kosten sind viel zu gross. Verwendet man zum Beispiel das Runge-Kutta-Verfahren nach Engle mit einer Toleranz $TOL = 10^{-3}$, so werden 2994 Funktionsauswertungen der rechten Seite der Differentialgleichung benötigt. Bei einem modernen Programm für steife Differentialgleichungen benötigt man aber nur 75 solche Auswertungen. Um zu verstehen, warum das eine Verfahren viel grössere Schwierigkeiten hat als das andere genügt es, das entkoppelte Gleichungssystem

$$(5.88) \quad \begin{aligned} u' &= \lambda_1 u \\ v' &= \lambda_2 v \end{aligned}$$

mit demselben Verfahren und mit derselben Schrittweite zu lösen. Hierbei sind

$\lambda_1 = -199.21567416$ und $\lambda_2 = -0.78432584$ die Eigenwerte der Matrix

$$\begin{pmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -156.25 & -200 \end{pmatrix} .$$

Diese Eigenwerte geben gerade die Dämpfungen des Systems an, wie man aus der exakten Lösung von (5.86)

$$(5.89) \quad y(t) = 1 + 0.49e^{\lambda_1 t} + 3.26e^{\lambda_2 t} + 0.316 \cos(t - 0.66)$$

leicht sieht (siehe Fig. 5.11). Wir lösen (5.88) mit einem expliziten Verfahren. Der Einfachheit halber wählen wir das Euler-Verfahren.

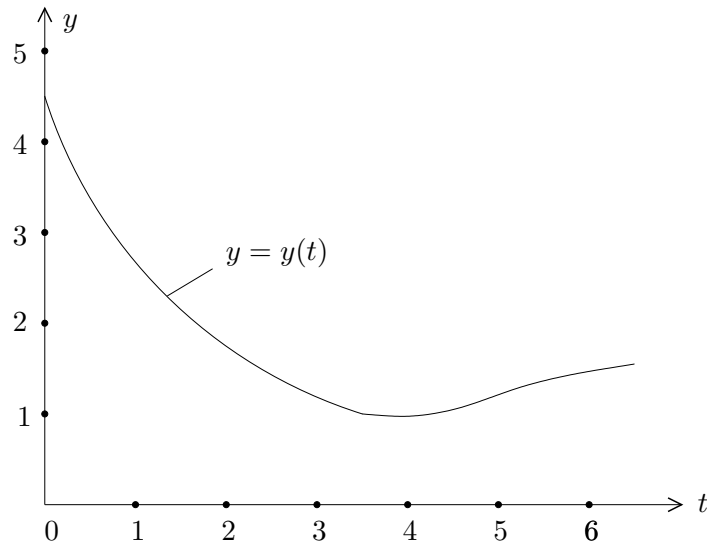


Fig. 5.11

Wir erhalten

$$(5.90) \quad \begin{aligned} u_{n+1} &= u_n + h\lambda_1 u_n = (1 + h\lambda_1)u_n = (1 + h\lambda_1)^{n+1}u_0 \\ v_{n+1} &= v_n + h\lambda_2 v_n = (1 + h\lambda_2)v_n = (1 + h\lambda_2)^{n+1}v_0 \end{aligned}$$

Damit diese Lösung nicht ins Unbegrenzte wächst, muss $|1 + h\lambda_1| \leq 1$ und $|1 + h\lambda_2| \leq 1$ gelten. Dies ist aber nur der Fall, wenn $-2 \leq h\lambda_1 \leq 0$ und $-2 \leq h\lambda_2 \leq 0$ erfüllt ist. Man erkennt sofort, dass wegen des stark gedämpften Terms mit $\lambda_1 \approx -199.2$ $h \leq 0.010039$ gelten muss. Dies ist sogar der Fall, wenn der Beitrag dieses Terms in der exakten Lösung unter die Maschinengenauigkeit abgesunken ist. Dies ist bei einem Rechner mit der

Maschinengenauigkeit von $9 \cdot 10^{-19}$ bereits bei $t \geq 0.21$ der Fall. Danach muss aber, obwohl von diesem Eigenwert kein Beitrag mehr in den Lösungen sichtbar ist, der Integrationsschritt h künstlich klein gehalten werden.

Dieses Phänomen heisst **Steifheit der Differentialgleichungen**. Man beachte, dass diese Erscheinung nicht etwa durch Instabilität des mechanischen Systems hervorgerufen wird. Vielmehr gilt, je stabiler das mechanische System ist, je stärker die Dämpfung ist, umso kleiner muss die Schrittweite gewählt werden. Würde man zum Beispiel beim freien Fall die Kugel an einen Fallschirm hängen (Beispiel 2 aus 5.1), so wird das Problem durch die starke Reibung steif gemacht, obwohl der Fall gebremst wird. Es hilft auch nichts, wenn man eine genauere Rechenmaschine verwendet, denn die Bedingungen $|1 + h\lambda_1| \leq 1$ und $|1 + h\lambda_2| \leq 1$ sind unabhängig von der Rechnergenauigkeit. Man beachte, dass im Falle eines nicht stark gedämpften Schwingers die Eigenwerte λ_1 und λ_2 konjugiert komplex werden. Auch in diesem Falle liegt Steifheit vor, wenn $\gamma \gg 1$ ist. Ganz allgemein heisst ein Differentialgleichungssystem steif, wenn die Realteile der Eigenwerte der Jacobi-Matrix $\frac{\partial f}{\partial y}$ negativ oder zumindest klein sind und einige dieser Realteile sehr stark negativ sind. Solche Systeme treten zum Beispiel immer dann auf, wenn starke Dämpfungen vorliegen. Weitere Beispiele liefern chemische Reaktionsvorgänge, wo die einzelnen Prozesse mit sehr unterschiedlichen Reaktionsgeschwindigkeiten ablaufen. Es gibt zum Beispiel Ionen-Reaktionsabläufe in der Erdatmosphäre mit über 50 verschiedenen Ionenarten und Unterschieden in der Reaktionsgeschwindigkeit von mehr als 10^7 . Beim Lösen von Grenzschichtproblemen treten häufig steife Systeme auf. Wird zur Berechnung eines Wärmeleitungs- oder eines Diffusionsprozesses zunächst eine Diskretisierung in Raumrichtung vorgenommen, so entstehen steife Probleme, wobei alle Eigenwerte negativ reell sind und einige beliebig stark negativ werden, wenn das Diskretisierungsgitter in Raumrichtung verfeinert wird.

Wir wollen uns überlegen welche Verfahren zum Lösen steifer Differentialgleichungen verwendet werden sollen. Im obigen Beispiel haben wir, um das Verhalten des Euler-Verfahrens zu studieren, die nichtlineare rechte Seite von (5.86) durch eine lineare entkoppelte rechte Seite (5.88) ersetzt. Hierbei waren λ_1 und λ_2 die Eigenwerte der Jacobimatrix der rechten Seite von (5.86). Im allgemeinen Fall eines Systems von m Gleichungen werden wir analog vorgehen. Anstelle ein Verfahren auf das System (5.16)

$$(5.91) \quad \frac{d\mathbf{y}(t)}{dt} = \mathbf{f}(t, \mathbf{y}(t))$$

anzuwenden, wenden wir es auf das folgende entkoppelte System an:

$$(5.92) \quad \begin{aligned} \frac{dy_1(t)}{dt} &= \lambda_1 y_1(t) \\ \frac{dy_2(t)}{dt} &= \lambda_2 y_2(t) \\ &\vdots \\ \frac{dy_m}{dt} &= \lambda_m y_m(t) \end{aligned}$$

Hierbei sind $\lambda_1, \dots, \lambda_m$ die Eigenwerte der Jacobimatrix $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t, \mathbf{y}(t))$. Zudem nehmen wir vereinfachend an, diese Matrix sei diagonalisierbar. Da das System (5.92) entkoppelt ist, und jede Gleichung vom Typ

$$(5.93) \quad z'(t) = \lambda z(t)$$

ist, genügt es, das Verhalten des Verfahrens an dieser Gleichung zu studieren. Gibt man die Anfangsbedingung

$$z(0) := z_0$$

vor, so ist

$$(5.94) \quad z(t) = e^{\lambda t} z_0$$

die exakte Lösung. Da λ ein Eigenwert von $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$ ist, kann λ auch als komplexe Zahl angenommen werden. Mit

$$\lambda = Re\lambda + i Im\lambda = Re\lambda + i\omega$$

lässt sich für $\omega = Im\lambda \neq 0$ die Lösung (5.94) auch schreiben als

$$\begin{aligned} z(t) &= e^{Re\lambda t} \cdot e^{i\omega t} z_0 \\ &= e^{Re\lambda t} (\cos \omega t + i \sin \omega t) z_0. \end{aligned}$$

Dies bedeutet, dass die Lösung

- ein exponentielles Wachstum besitzt, falls $Re\lambda > 0$,
- eine exponentielle Dämpfung besitzt, falls $Re\lambda < 0$.

Ist $\omega \neq 0$, so hat man zusätzlich eine Oszillation mit der Frequenz $\omega/(2\pi)$.

Wir wenden als Beispiel das sogenannte θ -Verfahren auf die Gleichung (5.94) an. Dies ist ein Einschrittverfahren und ist für $y' = f(t, y)$ wie folgt definiert:

$$(5.95) \quad y_{n+1} = y_n + h((1 - \theta) f(t_n, y_n) + \theta f(t_{n+1}, y_{n+1})).$$

Offenbar ist das Verfahren implizit, falls $\theta \neq 0$ ist, d.h. (5.95) stellt eine im allgemeinen nichtlineare Gleichung für y_{n+1} dar. (Für Lösungsmethoden zur Berechnung von y_{n+1} siehe Abschnitt 5.4 D). Ist $\theta = 0$, so ist dies das Euler-Verfahren. Ist $\theta = 1$, so erhalten wir das sogenannte implizite Euler-Verfahren (siehe auch Beispiel 5). $\theta = \frac{1}{2}$ ergibt die Trapezregel, siehe Beispiel (5.73). Anwenden auf (5.93) liefert

$$\begin{aligned} z_{n+1} &= z_n + h(1 - \theta)\lambda z_n + h\theta\lambda z_{n+1} \\ (1 - h\theta\lambda)z_{n+1} &= (1 + h(1 - \theta)\lambda)z_n \end{aligned}$$

und mit der Abkürzung

$$(5.96) \quad h\lambda = \mu \in \mathbb{C}$$

ergibt sich

$$z_{n+1} = \frac{1 + (1 - \theta)\mu}{1 - \theta\mu} z_n.$$

Man erkennt sofort, dass die numerische Lösung für festes h , und damit festes μ , exponentiell anwächst, falls gilt

$$\left| \frac{1 + (1 - \theta)\mu}{1 - \theta\mu} \right| > 1,$$

respektive beschränkt bleibt, falls gilt

$$(5.97) \quad \left| \frac{1 + (1 - \theta)\mu}{1 - \theta\mu} \right| \leq 1.$$

Numerisch ist man am Fall (5.97) interessiert und nennt die Menge aller μ für die (5.97) gilt das Stabilitätsgebiet des θ -Verfahrens. Allgemein heisst $S \subset \mathbb{C}$ das Stabilitätsgebiet eines Verfahrens, falls gilt

$$S = \{h\lambda \in \mathbb{C} \mid |z_n| \text{ beschränkt für } n \in \mathbb{N}\},$$

wobei z_n erhalten wird durch Anwenden des Verfahrens auf (5.93) mit konstanter Schrittweite h . Für die drei Spezialfälle des θ -Verfahrens $\theta = 0, \frac{1}{2}, 1$ erhalten wir:

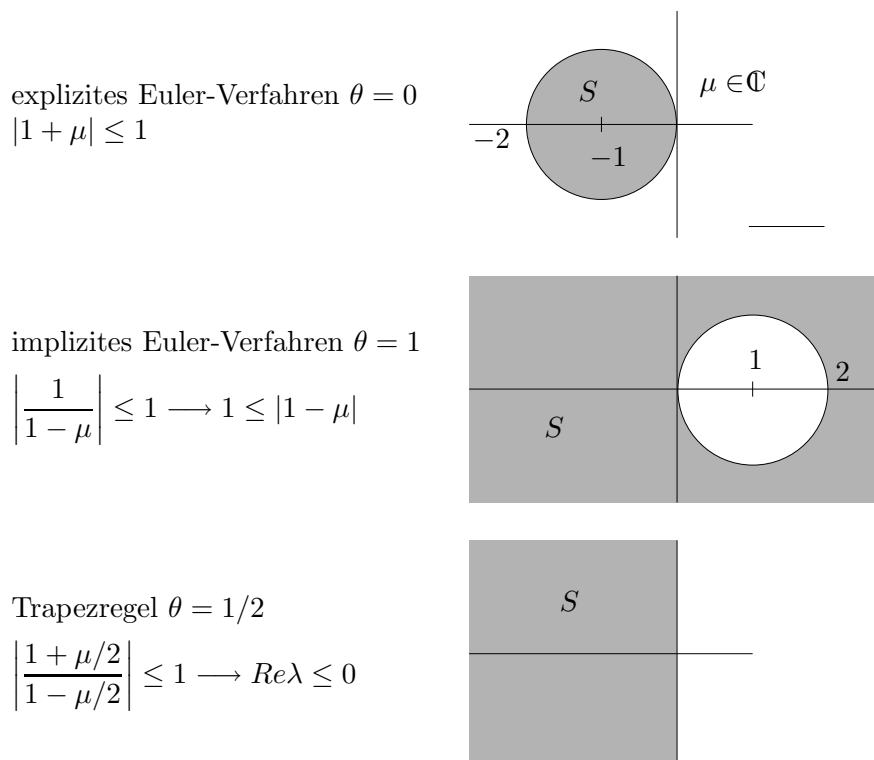


Fig. 5.13: Stabilitätsgebiete des θ -Verfahrens für $\theta = 0, \theta = 1, \theta = \frac{1}{2}$

Für die Trapezregel gilt, dass die numerische Lösung genau dann beschränkt ist, wenn dies auch für die exakte Lösung gilt. Wir erkennen, dass das explizite Euler-Verfahren ein beschränktes Stabilitätsgebiet hat. Dies ist typisch für alle expliziten Verfahren. Generell haben explizite Verfahren ein Stabilitätsgebiet der in Fig. 14 angegebenen Form

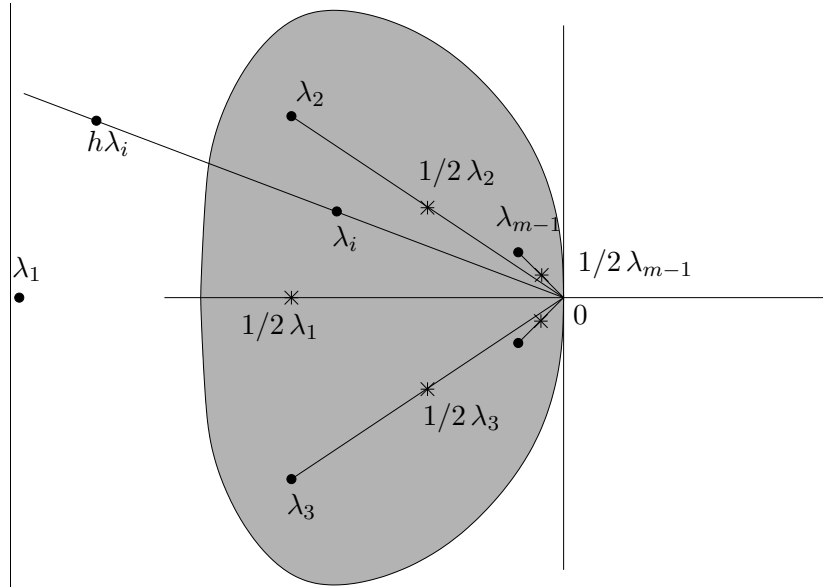


Fig. 5.14: Typisches Stabilitätsgebiet eines expliziten Verfahrens

Die Jacobimatrix des Systems besitze die Eigenwerte $\lambda_1, \lambda_2, \dots, \lambda_m$ mit

$$Re\lambda_1 \leq Re\lambda_2 \leq \dots \leq Re\lambda_{m-1} \leq Re\lambda_m \lesssim 0.$$

Wird für die Schrittweite zum Beispiel $h = 1$ gewählt, so liegt $\mu_1 = 1 \cdot \lambda_1$ nicht in S und somit ist der zugehörige Faktor grösser 1 und die numerische Lösung wächst exponentiell an, obwohl die zugehörige exakte Komponente jene ist, die am schnellsten abfällt. Würde $h = \frac{1}{2}$ gewählt, so liegen alle $1/2\lambda_i$ in S und man kann gefahrlos integrieren. Man erkennt leicht, dass sich jeweils $h\lambda_i$ auf einem Strahl durch den Ursprung 0 bewegt, wenn h verändert wird. Offenbar verlässt $h\lambda_i$ das Stabilitätsgebiet nicht, falls $Re\lambda_i \leq 0$ ist und man entweder das implizite Euler-Verfahren oder die Trapezregel benutzt. Bei diesen Verfahren muss man also den Schritt wegen einer unwichtigen Komponente **nicht** verkleinern. Deshalb sind sie gut geeignet für zum Lösen steifer Differentialgleichungen. Dasselbe gilt für die Rückwärtsdifferenzformeln, Abschnitt 5.4 C der Ordnung 1, 2, ... 6. Dies erkennt man aus deren Stabilitätsgebieten, siehe Anhang.

Für steife Differentialgleichungen sollte ein Programm verwendet werden, das auf Rückwärtsdifferenzformeln beruht, es sei denn, $\frac{\partial f}{\partial y}$ habe Eigenwerte nahe bei der imaginären Achse. Ist dies der Fall, so muss auf die hier nicht besprochenen adaptiven Verfahren (die Näherungen von $\frac{\partial f}{\partial y}$ verwenden) oder impliziten Runge-Kutta-Verfahren zurückgegriffen werden.

5.6 Zur Praxis der Anwendung von Programmen

Die Hauptunterscheidung, die gemacht werden muss, liegt zwischen steifen und nicht-steifen Differentialgleichungen.

A. Nicht-steife Differentialgleichungen

Bei diesen kann man vor allem explizite Verfahren verwenden. Die drei üblichen Typen sind explizite Runge-Rutta-Verfahren, z.B. Dormand-Prince Formeln 5(4), respektive 8(7), Prädiktor-Korrektor Adams-Moulton-Verfahren, und Extrapolationsverfahren. Die numerischen Resultate hängen auch von der Implementation im Rechenprogramm ab. Wir zitieren hier Ergebnisse aus sechs Beispielen aus dem Buch E. Hairer, S.P. Norsett, G. Wanner, Solving Ordinary Differential Equations I, Nonstiff Problem, Springer, 1987. Die verwendeten Programme basieren auf folgenden Verfahren:

DOPRI8	Runge-Kutta Dormand-Prince 8(7)
ODEX	Extrapolation Implementation Hairer, Norsett, Wanner
DIFEX1	Extrapolation Implementation Deuffhard
DEABM	Adams-Bashforth Moulton Formeln (nach Shampine-Gordon und RWTH Aachen)
EPISODE	Nordsieck-Representation of Adams Formeln (Implementation Byrne-Hindmarsh)
LSODE	Adams methods, da MF = 10 (Hindmarsh)
D02CAF	NAG-subroutine für Adams Formeln

In den Diagrammen wird die Anzahl der Auswertungen der rechten Seite ($fc =$ function calls = Mass für den Rechenaufwand) gegenüber dem negativen Logarithmus der im Programm angegebenen Toleranzgrenze für den lokalen Fehler angegeben. Da der Rechenaufwand steigen sollte, wenn man die Toleranz verkleinert, sollte ein ideales Programm stets eine monoton wachsende Funktion ergeben.

Ein Programm ist zudem für eine gegebene Toleranz besser als ein anderes, falls es für diese Toleranz weniger Funktionsauswertungen benötigt. In all den folgenden Beispielen aus dem oben erwähnten Buch, siehe Seite 169, ist das grundsätzliche Verhalten der Diagramme wie in Fig. 5.12 angegeben.

Man erkennt, dass für glatte Probleme bei Toleranzen zwischen 10^{-3} und 10^{-12} das Programm DOPRI8 verwendet werden soll. Will man härtere Toleranzen als 10^{-12} fordern, so sind die Extrapolationsverfahren wegen ihrer variablen Fehlerordnung besser.

Im folgenden vergleichen wir Runge-Kutta-Verfahren mit den verschiedenen Implementierungen der Adams Codes. Sieht man sich die folgenden Diagramme an, bei denen der Rechenaufwand mit Hilfe der Anzahl Funktionsauswertungen fc der rechten Seite gemessen wird, so sieht man, dass in der Regel das Runge-Kutta-Programm DOPRI8 schlechter abschneidet (siehe Figur auf Seite 170).

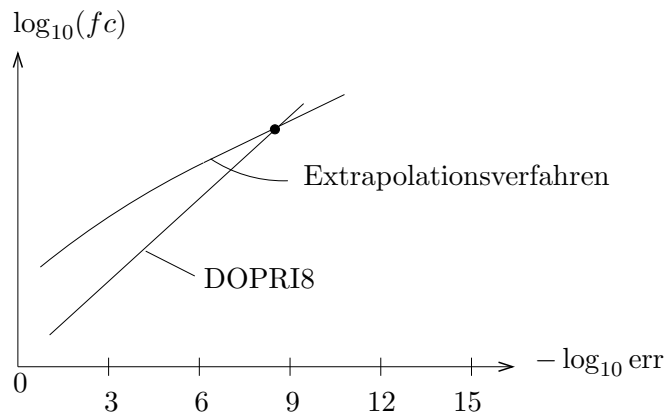


Fig. 5.15

Benutzt man aber das Mass der Rechenzeit zum Vergleich, so schneidet das Runge-Kutta-Verfahren DOPRI8 besser ab. Dies liegt daran, dass die verwendeten Beispiele alle relativ kleine Systeme sind, $m \leq 10$ und die Auswertung schnell geschieht. Damit folgt, dass man bei grossen Systemen, z.B. $m \geq 10'000$, oder bei Systemen mit aufwendiger rechten Seite, Adams-Moulton basierte Programme verwendet werden sollte. Sind die Systeme klein, resp. die rechte Seite schnell auswertbar, so sollen Runge-Kutta basierte Programme, z.B. DOPRI8, verwendet werden.

Der benötigte Speicherplatz für die angegebenen Programme ist in der folgenden Tabelle angegeben.

Speicherplatz für System mit m Komponenten

DEABM	22 m
EPISODE	18 m
LSODE	17 m
D02CAF	19 m
DOPRI8	9 m

Neuere numerische Resultate findet man in der zweiten Ausgabe des oben erwähnten Buches.

B. Steife Differentialgleichungen

In diesem Abschnitt stützen wir uns auf die numerischen Resultate aus dem Buch E. Hairer, G. Wanner, Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems, 2nd Edition, Springer 1996. Die verwendeten Programme basieren auf folgenden Verfahren:

ROS4, RODAS	Sogenannte Rosenbrock-Verfahren (dies sind modifizierte Runge-Kutta-Verfahren)
SDIRK, RADAU5, STRIDE	Dies sind implizite Runge-Kutta-Verfahren
SEULEX	linear implizites Eulerverfahren mit Extrapolation
SODEX	steife Version der Bader-Deuffhard-Extrapolation
LSODE	mit MF = 21,22,24 oder 25 Nordsieck-Form der Rückwärts-Differentiationsformeln
DEBDF	Shampine-Watts-Modifikation von LSODE
VODE	Variable Koeffizienten-Verfahren basierend auf BDF-Formeln
SPRINT	'Blended' Mehrschrittverfahren von Skeel und Kong
SECDER	Mehrschrittverfahren die die zweite Ableitung verwenden

Die Beispiele, die verwendet werden, haben die folgende Systemgrösse m und sind im oben erwähnten Buch explizit beschrieben.

		m
VPPOL	Van der Pol Gleichung	2
ROBER	Robertson Problem	3
OREGO	Oregonator	3
HIRES	physiologisches Problem	8
E5	schlecht skaliertes chemisches Problem	4
PLATE	Auto auf einer Platte fahrend	80
BEAM		40
CUSP	Cusp Katastrophe	96
BRUSS	Brusselator	500
KS	Kuramoto-Sivashinsky Gleichung	1022

Die folgenden Ergebnisse stammen aus dem oben erwähnten Buch (siehe nachfolgende Seiten 171-176).

Rosenbrock Programme sind gut für grobe Toleranzen zwischen 10^{-3} und 10^{-5} . Die Extrapolation, vorallem SEULEX, ist aber gut für kleine Toleranzen, d.h. 10^{-9} . LSODE ist praktisch stets sehr schnell, ausser beim Beispiel BEAM. Dies liegt dort daran, dass es nicht auf A -stabilen Formeln beruht. DEBDF verhält sich etwa wie LSODE, ist aber ab und zu etwas schneller. VODE verhält sich bei grossen Problemen etwa wie LSODE und DEBDF. SPRINT enthält A -stabile Formeln bis zur Ordnung 4 und ist deshalb auf dem Problem BEAM besser als LSODE. Da SECDER die zweite Ableitung benötigt, ist es nicht für grosse Probleme geeignet.

Für eine genauere Beschreibung der Probleme, der Eigenschaften der Programme und der Ergebnisse, konsultiere man oben zitiertes Buch.