

8. RUNGE-KUTTA-VERFAHREN

In diesem Kapitel suchen wir nach Näherungen für die Lösung $y : [t_0, T] \rightarrow \mathbb{R}^d$ der Anfangswertaufgabe

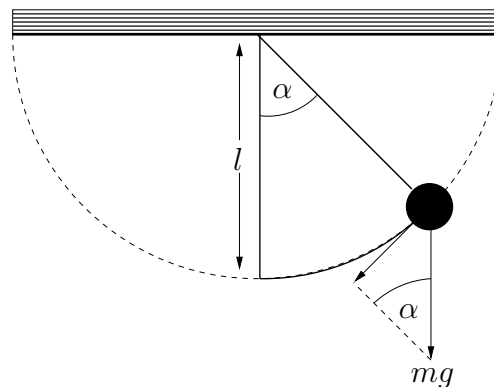
$$\begin{aligned} y'(t) &= f(t, y(t)), & t \in [t_0, T] \\ y(t_0) &= y_0. \end{aligned} \quad (8.1)$$

Hierbei sei $f : U \rightarrow \mathbb{R}^d$, wobei $U \subset \mathbb{R} \times \mathbb{R}^d$ eine offene und zusammenhängende Menge ist und $(t_0, y_0) \in U$.

8.1 Beispiele von Differentialgleichungen

Physik

Bewegung eines Pendels:



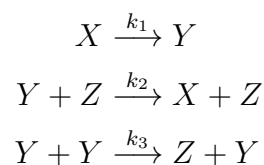
$$\begin{aligned} ms''(t) &= -mg \sin \alpha(t), & s &= \ell \alpha, \\ \alpha''(t) &= -\frac{g}{\ell} \sin \alpha(t). \end{aligned}$$

Durch Setzen von $y = [\alpha \ \alpha']^T$ zu einer gegebenen Ausgangslage $\alpha(0)$ und einer gegebenen Anfangsgeschwindigkeit $\alpha'(0)$ erhält man ein System der Form (8.1).

Differentialgleichungen treten etwa bei der Berechnung von Satellitenbahnen, in der Robotik, bei der Simulation elektrischer Netzwerke, usw. auf.

Chemie

Hier möchte man den Verlauf chemischer Reaktionen simulieren. Weiß man zum Beispiel, dass die Substanzen X, Y, Z gemäß



mit Reaktionskonstanten k_1, k_2, k_3 reagieren, dann liefert das Massenwirkungsgesetz für die Konzentrationen $x(t), y(t), z(t)$ der Substanzen X, Y, Z

$$\begin{aligned}x' &= -k_1x + k_2yz, \\y' &= k_1x - k_2yz - k_3y^2, \\z' &= k_3y^2.\end{aligned}$$

Biologie

Die Anzahl $y(t)$ von Speisefischen zur Zeit t und die Anzahl $z(t)$ von Raubfischen kann mit Hilfe des Populationsmodells

$$\begin{aligned}y' &= ay - byz, \\z' &= -cz + dyz\end{aligned}$$

berechnet werden.

Weitere Anwendungen sind Epidemiemodelle, Herzschlag, usw.

Raumdiskretisierung partieller Differentialgleichungen

Ein wichtiges Beispiel für Anfangs-Randwertprobleme partieller Differentialgleichungen ist die Wärmeleitungsgleichung. Die Wärmeverteilung in einem eindimensionalen Stab genügt

$$\begin{aligned}\frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} + g(x, t), & x \in \Omega = [0, 1], \quad t \in [0, T] \\u(x, t) &= 0, & x \in \{0, 1\} & \text{Randbedingung} \\u(x, 0) &= u_0(x) & x \in \Omega & \text{Anfangsbedingung}\end{aligned}$$

Wir diskretisieren diese Gleichung bezüglich der Ortsvariablen x , d. h. wir lösen das Problem nicht mehr im Intervall $[0, 1]$ sondern nur noch auf diskreten Punkten $x_j = j\Delta x$, $j = 0, \dots, N$, mit der Ortsschrittweite $\Delta x = 1/N$. Hierzu ersetzen wir die Ableitung $\frac{\partial^2 u}{\partial x^2}$ durch einen zentralen Differenzenquotienten zweiter Ordnung:

$$\frac{\partial^2}{\partial x^2} u(x_j, t) \approx \frac{1}{(\Delta x)^2} (u(x_{j-1}, t) - 2u(x_j, t) + u(x_{j+1}, t)).$$

Wir bezeichnen mit $y_j(t)$, $j = 1, \dots, N-1$ die Approximation an $u(x_j, t)$. Dann ergibt sich durch die Ortsdiskretisierung ein System gewöhnlicher Differentialgleichungen der Form

$$\begin{aligned}y_j' &= \frac{1}{(\Delta x)^2} (y_{j-1} - 2y_j + y_{j+1}) + g(x_j, t), & j &= 1, \dots, N-1, \\y_j(0) &= u_0(x_j), & j &= 0, \dots, N, \\y_0(t) = y_N(t) &= 0, & t &\geq 0.\end{aligned}$$

In Kurzform kann man dieses System auch als

$$y'(t) = Ay(t) + g(t), \quad A = \frac{1}{(\Delta x)^2} \begin{bmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix}, \quad g(t) = \begin{bmatrix} g(x_1, t) \\ \vdots \\ \vdots \\ g(x_{N-1}, t) \end{bmatrix},$$

schreiben. Aus dem Satz von Gershgorin folgt, dass alle Eigenwerte von A im Intervall $[-\frac{4}{(\Delta x)^2}, 0]$ liegen.

8.2 Theorie gewöhnlicher Differentialgleichungen

Wir zeigen zunächst, dass es genügt **autonome** Differentialgleichungen erster Ordnung,

$$y' = f(y), \quad y(t_0) = y_0, \quad (8.2)$$

zu betrachten, denn Differentialgleichungen höherer Ordnung und nichtautonome Differentialgleichungen lassen sich darauf zurückführen.

Wie im ersten Beispiel des Pendels treten häufig Differentialgleichungen höherer Ordnung auf:

$$y^{(k)} = f(t, y, y', \dots, y^{(k-1)}).$$

Jede solche Differentialgleichung lässt sich überführen in ein System erster Ordnung. Wir erhalten nämlich aus

$$\begin{array}{ll} y_1 = y & y'_1 = y_2 \\ y_2 = y' & y'_2 = y_3 \\ \vdots & \vdots \\ y_{k-1} = y^{(k-2)} & y'_{k-1} = y_k \\ y_k = y^{(k-1)} & y'_k = f(t, y_1, \dots, y_k) \end{array}$$

mit $Y = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix}$ das System $Y' = F(t, Y)$.

Eine nichtautonome Differentialgleichung, das ist eine Differentialgleichung (8.1), bei der die rechte Seite f explizit von t abhängt, ist äquivalent zu einem autonomen System der Dimension $d + 1$:

$$\begin{array}{ll} t' = 1, & t(t_0) = t_0, \\ y' = f(t, y), & y(t_0) = y_0, \end{array}$$

denn für $Y = \begin{bmatrix} t \\ y \end{bmatrix} \in \mathbb{R}^{d+1}$ hat das System die Form $Y' = F(Y)$.

Als nächstes werden wir einige theoretische Resultate zur Existenz, Eindeutigkeit und Stabilität von Lösungen von (8.1) wiederholen. Dazu setzen wir im Folgenden voraus, dass $U \subset \mathbb{R} \times \mathbb{R}^d$ offen und zusammenhängend und $f : U \rightarrow \mathbb{R}^d$ stetig ist und eine lokale Lipschitz-Bedingung erfüllt, d. h. für $K \subset U$ kompakt gelte

$$\|f(t, y) - f(t, z)\| \leq L \|y - z\| \quad \forall (t, y), (t, z) \in K. \quad (8.3)$$

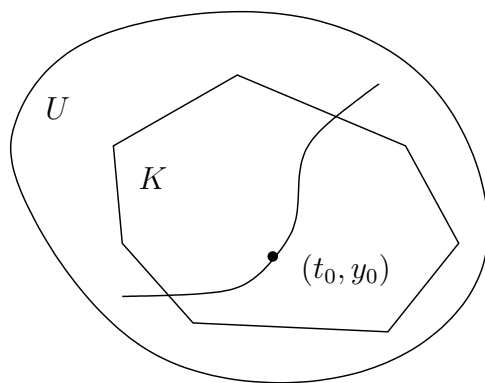
Die lokale Lipschitz-Bedingung (8.3) ist erfüllt, falls f stetig differenzierbar ist und zwar mit $L = \max_{(t,y) \in K} \|f_y(t, y)\|$.

Satz 8.1. (Lokale Existenz und Eindeutigkeit)

Unter obigen Voraussetzungen gibt es ein offenes Intervall I mit $t_0 \in I$ so, dass genau ein $y : I \rightarrow \mathbb{R}^d$ existiert, welches das Anfangswertproblem

$$y'(t) = f(t, y(t)), \quad \text{für } t \in I \quad \text{und} \quad y(t_0) = y_0$$

löst. Die Lösung kann bis an den Rand von U fortgesetzt werden, d. h. für alle $K \subset U$ kompakt mit $(t_0, y_0) \in K$ existiert ein $t_1 > t_0$ so, dass die Lösung des Anfangswertproblems auf $[t_0, t_1]$ existiert mit $(t_1, y(t_1)) \notin K$. \square



Beispiel. Die rechte Seite der Differentialgleichung

$$y' = y^2, \quad y(0) = 1, \quad U = \mathbb{R} \times \mathbb{R}.$$

ist lokal Lipschitz-stetig, erfüllt also die Voraussetzungen von Satz 8.1. Die eindeutige Lösung $y(t) = (1-t)^{-1}$ existiert auf dem offenen Intervall $I = (-\infty, 1)$ und $t_0 = 0 \in I$. Es gilt jedoch $\lim_{t \nearrow 1} y(t) = +\infty$. \diamond

Für numerische Verfahren ist es wichtig, wie sich Störungen der Anfangswerte auf die Lösung auswirken. Um dies zu untersuchen, nehmen wir für die rechte Seite der Differentialgleichung (8.1) an, dass $f : [t_0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ die *einseitige Lipschitz-Bedingung*

$$\langle f(t, y) - f(t, z), y - z \rangle \leq \ell \|y - z\|^2 \quad \text{für alle } y, z \in \mathbb{R}^d \quad (8.4)$$

für ein Skalarprodukt $\langle \cdot, \cdot \rangle$ auf \mathbb{R}^d und die dadurch induzierte Norm $\|\cdot\|$ erfüllt.

Bemerkung. Falls f Lipschitz-stetig mit Lipschitz-Konstante L ist, so ist (8.4) mit $\ell = L$ erfüllt, denn aus der Cauchy-Schwarz'schen Ungleichung folgt

$$|\langle f(t, y) - f(t, z), y - z \rangle| \leq \|f(t, y) - f(t, z)\| \|y - z\| \leq L \|y - z\|^2.$$

Für das kleinstmögliche ℓ kann jedoch $\ell \ll L$ gelten. Dieser Fall ist typisch für steife Differentialgleichungen. Es ist $\ell < 0$ möglich, wogegen immer $L \geq 0$ gilt.

Satz 8.2. Für f sei (8.4) erfüllt. $y, z : [t_0, T] \rightarrow \mathbb{R}^d$ seien Lösungen der Differentialgleichung (8.1) zu den Anfangswerten $y(t_0) = y_0, z(t_0) = z_0$. Dann gilt

$$\|y(t) - z(t)\| \leq e^{\ell(t-t_0)} \|y_0 - z_0\| \quad \text{für alle } t \in [t_0, T].$$

Beweis. Es gilt

$$\begin{aligned} \frac{d}{dt} \|y(t) - z(t)\|^2 &= 2 \langle y'(t) - z'(t), y(t) - z(t) \rangle \\ &= 2 \langle f(t, y(t)) - f(t, z(t)), y(t) - z(t) \rangle \\ &\leq 2\ell \|y(t) - z(t)\|^2. \end{aligned}$$

Falls $y(t_0) \neq z(t_0)$ gilt, so ist wegen der Eindeutigkeit der Lösung von (8.1) auch $y(t) \neq z(t)$ für alle t . Damit ist

$$\varphi(t) = \|y(t) - z(t)\|^2 \neq 0$$

für alle t und wir erhalten aus obiger Relation

$$\frac{\varphi'(t)}{\varphi(t)} = \frac{d}{dt} \log \varphi(t) \leq 2\ell.$$

Durch Integration dieser Ungleichung ergibt sich

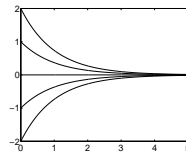
$$\log \varphi(t) - \log \varphi(t_0) \leq 2\ell(t - t_0)$$

oder $\varphi(t) \leq e^{2\ell(t-t_0)}\varphi(t_0)$. □

Der Fehler in den Anfangsdaten $y_0 = y(t_0)$ kann sich also mit dem Faktor $e^{\ell(t-t_0)}$ verstärken.

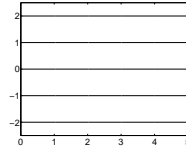
Beispiel. Die Gleichung $y' = \lambda y$ hat die Lösung $y(t) = y(t_0)e^{\lambda(t-t_0)}$. Die einseitige Lipschitz-Konstante ist $\ell = \lambda$, die Lipschitz-Konstante $L = |\lambda|$.

$\lambda < 0$: Fehler dämpfen sich aus



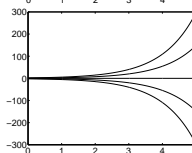
“asymptotisch stabil”

$\lambda = 0$: keine Fehlerverstärkung



“stabil”

$\lambda > 0$: Fehlerverstärkung

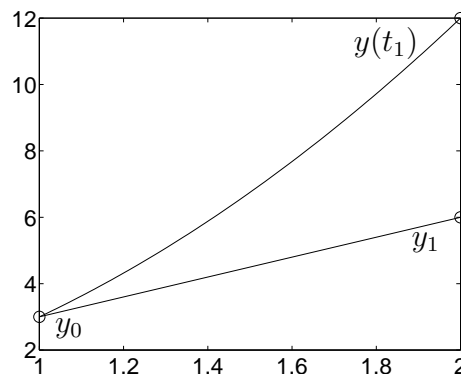


“instabil”

Wir werden später diese Testgleichung immer wieder zum Vergleich numerischer Verfahren heranziehen. Von guten numerischen Verfahren erwartet man, dass die numerische Lösung sich ähnlich verhält wie die exakte Lösung. \diamond

8.3 Euler-Verfahren

Das einfachste und älteste Verfahren zur näherungsweisen Lösung von (8.1) ist das Euler-Verfahren (L. Euler, 1768). Die Idee ist, lokal die (unbekannte) Lösung durch die (bekannte) Tangente zu ersetzen. Ausgehend von $y_0 = y(t_0)$ erhält man an der Stelle $t_1 = t_0 + h$

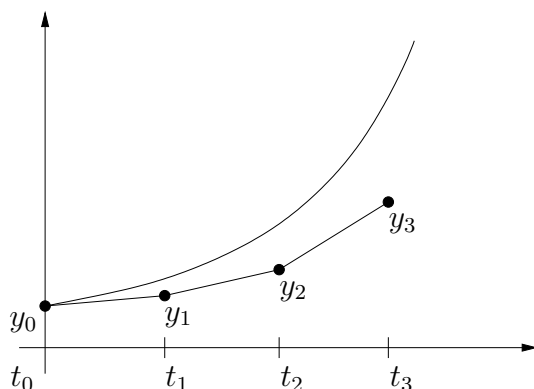


Hieraus ergibt sich

$$y_1 = y_0 + hf(t_0, y_0)$$

als Approximation an $y(t_1)$. Man nimmt dann y_1 als Startwert für den nächsten Schritt, d. h. man setzt

$$\begin{aligned} y_2 &= y_1 + hf(t_1, y_1), & t_2 &= t_1 + h = t_0 + 2h, \\ \vdots & & \vdots & \\ y_{n+1} &= y_n + hf(t_n, y_n), & t_n &= t_0 + nh. \end{aligned}$$



Jetzt muss man sich natürlich fragen, ob y_n eine Approximation an $y(t_n)$ ist, falls $h \rightarrow 0$?

Wir nehmen dazu an, dass $I = [t_0, T]$ ein Intervall ist, $f : I \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ stetig differenzierbar und (global) Lipschitz-stetig ist, also

$$\|f(t, y) - f(t, z)\| \leq L\|y - z\| \quad \forall y, z \in \mathbb{R}^d, \quad \forall t \in I, \quad (8.5)$$

wobei

$$L = \sup_{(t,y) \in I \times \mathbb{R}^d} \left\| \frac{\partial f}{\partial y}(t, y) \right\| < \infty.$$

Ist $y : I \rightarrow \mathbb{R}^d$ Lösung von (8.1), dann ist y nach Voraussetzung zweimal stetig differenzierbar, denn

$$y'' = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y} y' = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y} f.$$

Die Folge y_n sei für $n = 0, 1, 2, \dots$ mit $t_n = t_0 + nh \in I$ wie oben durch das Euler-Verfahren definiert.

Satz 8.3. *Unter obigen Voraussetzungen gilt für den Fehler des Euler-Verfahrens*

$$\|y_n - y(t_n)\| \leq Mh,$$

wobei

$$M = \frac{e^{L(T-t_0)} - 1}{L} \frac{1}{2} \max_{t \in I} \|y''(t)\|.$$

Insbesondere gilt

$$\lim_{h \rightarrow 0} \max_{n: t_n \in I} \|y_n - y(t_n)\| = 0,$$

d. h. die Näherungslösung konvergiert gleichmäßig gegen die exakte Lösung der Anfangswertaufgabe, wenn die Schrittweite gegen Null geht.

Beweis. Wir beweisen die Aussage in drei Schritten.

- (a) Lokaler Fehler: Wir untersuchen zunächst den Fehler nach einem Schritt des Verfahrens, wenn man auf der exakten Lösung $(t_n, y(t_n))$ startet. Der Satz von Taylor mit Integralrestglied liefert für die exakte Lösung

$$\begin{aligned} y(t_{n+1}) &= y(t_n) + hy'(t_n) + h^2 \int_0^1 (1-\theta)y''(t_n + \theta h)d\theta \\ &= y(t_n) + hf(t_n, y(t_n)) + d_{n+1}, \end{aligned}$$

wobei

$$\|d_{n+1}\| \leq \frac{1}{2}h^2 \max_{t \in [t_0, T]} \|y''(t)\| =: Ch^2.$$

- (b) Fehlerfortpflanzung: Ausgehend von den Anfangswerten v_n bzw. w_n ergeben sich durch einen expliziten Eulerschritt die Näherungen

$$\begin{aligned} v_{n+1} &= v_n + hf(t_n, v_n), \\ w_{n+1} &= w_n + hf(t_n, w_n). \end{aligned}$$

Subtraktion liefert wegen der Lipschitz-Bedingung

$$\begin{aligned} \|v_{n+1} - w_{n+1}\| &\leq \|v_n - w_n\| + h\|f(t_n, v_n) - f(t_n, w_n)\| \\ &\leq (1 + Lh)\|v_n - w_n\|. \end{aligned}$$

- (c) Fehlerakkumulation: Hierzu betrachten wir Lady Windermere's Fächer in Abbildung 8.1. Wir bezeichnen mit y_j^k die Näherung an $y(t_j)$ zum Anfangswert $y(t_k)$ nach $(j - k)$ Schritten, also insbesondere $y_n^0 = y_n$ und $y_n^n = y(t_n)$. Nach (a) gilt

$$\|y_{k+1}^k - y_{k+1}^{k+1}\| \leq Ch^2$$

und nach (b) ist

$$\begin{aligned} \|y_n^k - y_n^{k+1}\| &\leq (1 + hL)\|y_{n-1}^k - y_{n-1}^{k+1}\| \\ &\leq \dots \\ &\leq (1 + hL)^{n-k-1}\|y_{k+1}^k - y_{k+1}^{k+1}\| \\ &\leq (1 + hL)^{n-k-1}Ch^2. \end{aligned}$$

Ferner gilt

$$\begin{aligned} \|y_n - y(t_n)\| &= \|y_n^0 - y_n^n\| \\ &\leq \|y_n^0 - y_n^1\| + \|y_n^1 - y_n^2\| + \dots + \|y_n^{n-1} - y_n^n\| \\ &\leq Ch^2(1 + hL)^{n-1} + Ch^2(1 + hL)^{n-2} + \dots + Ch^2 \\ &= Ch^2 \frac{(1 + hL)^n - 1}{1 + hL - 1} \\ &\leq Ch \frac{e^{nhL} - 1}{L} \\ &\leq Ch \frac{e^{(T-t_0)L} - 1}{L}. \end{aligned}$$

Die letzte Ungleichung folgt aus $nh \leq T - t_0$. □

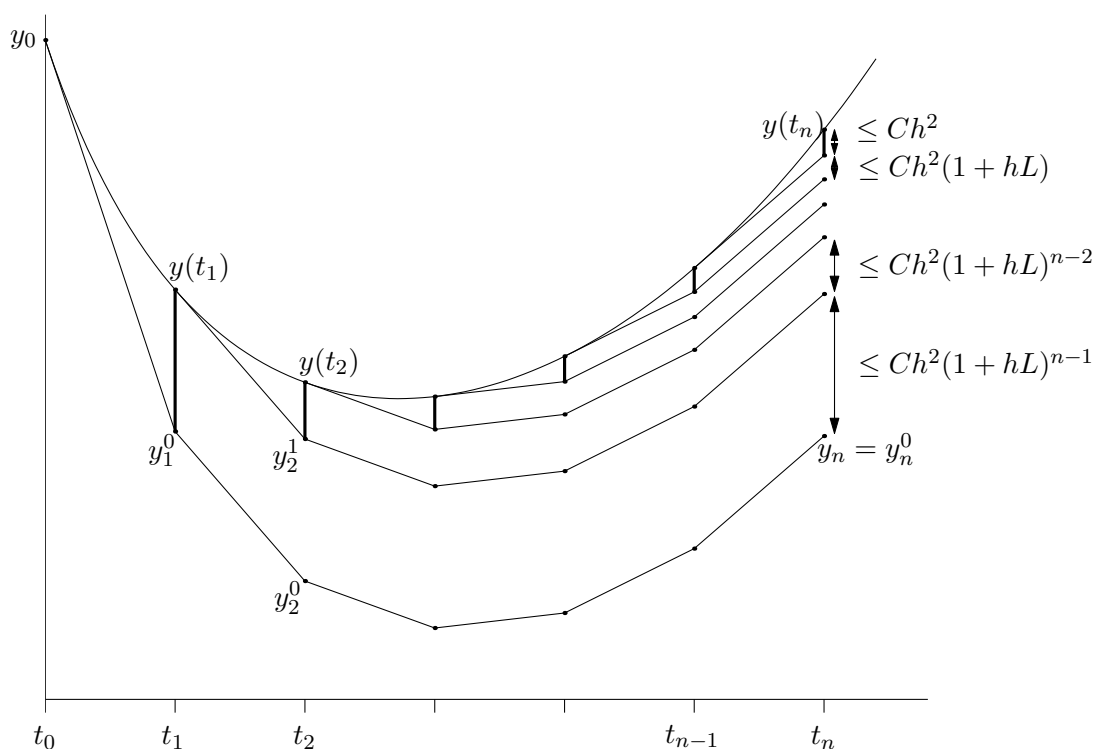
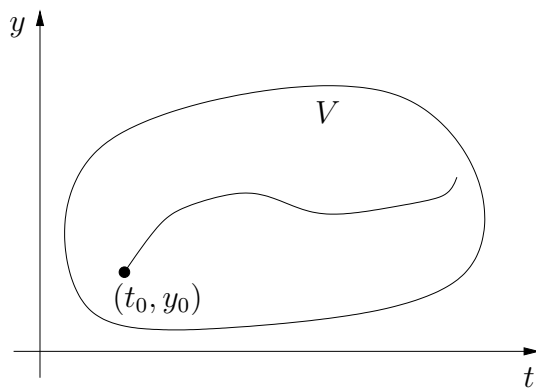


Abb. 8.1: Lady Windermere's Fächer

Bemerkung. Sei V eine beschränkte Umgebung der Trajektorie $\{(t, y(t)) \mid t \in I\}$.



Satz 8.3 zeigt: Falls die Schrittweite h genügend klein ist, so bleibt die numerische Trajektorie $\{(t_n, y_n) \mid t_n \in I\}$ in V . Insbesondere geht dann nur die Lipschitz-Konstante von $f|_V$ in die Fehlerabschätzung ein. Auf die globale Lipschitz-Bedingung kann daher verzichtet werden.

Fehler können beim expliziten Euler-Verfahren um den Faktor $(1+hL)$ verstärkt werden. Im Beispiel $y' = \lambda y$ in Abschnitt 8.1 haben wir jedoch gesehen, dass Fehler in den Anfangsdaten gedämpft werden, falls $\lambda < 0$.

Wenden wir das explizite Euler-Verfahren auf die Testgleichung an, so ergibt sich nach einem Schritt ausgehend von den Startwerten y_n, z_n

$$\begin{aligned} y_{n+1} &= y_n + h\lambda y_n, \\ z_{n+1} &= z_n + h\lambda z_n, \end{aligned}$$

also

$$\|y_{n+1} - z_{n+1}\| = |1 + h\lambda| \|y_n - z_n\|.$$

Für $|1 + h\lambda| > 1$ ist das explizite Euler-Verfahren instabil. Euler-Verfahren und exakte Lösung haben damit unterschiedliches Stabilitätsverhalten. Um ein stabiles Verfahren zu erhalten, muss man entweder die Schrittweite h so klein wählen, dass $|1 + h\lambda| \leq 1$ gilt oder nach Verfahren suchen, die ähnliches Stabilitätsverhalten wie die exakte Lösung aufweisen. Da die Verkleinerung von h für betragsmäßig große λ (die Differentialgleichung heißt dann **steif**) zu einem hohen Rechenaufwand und, wie wir im nächsten Abschnitt zeigen werden, zu großen Rundungsfehlern führt, werden wir später den zweiten Weg einschlagen und sogenannte implizite Verfahren behandeln. An dieser Stelle sei nur das implizite Euler-Verfahren erwähnt:

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}), \quad n = 0, 1, \dots$$

Im Allgemeinen erfordert das implizite Euler-Verfahren in jedem Zeitschritt die Lösung eines nichtlinearen Gleichungssystems, was einen hohen Rechenaufwand erfordern kann. In unseren linearen Testproblem lässt sich y_{n+1} jedoch explizit berechnen:

$$y_{n+1} = y_n + h\lambda y_{n+1} \iff y_{n+1} = \frac{1}{1 - h\lambda} y_n.$$

Betrachten wir wieder zwei Näherungen ausgehend von y_n und z_n und bilden die Differenz, so ergibt sich

$$\|y_{n+1} - z_{n+1}\| = \frac{1}{|1 - h\lambda|} \|y_n - z_n\|.$$

Das implizite Euler-Verfahren ist also für $|1 - h\lambda| \geq 1$ stabil, insbesondere für alle $\lambda < 0$. Im Gegensatz zum expliziten Euler-Verfahren haben hier numerische und exakte Lösung das gleiche Stabilitätsverhalten.

8.4 Einfluss von Rundungsfehlern

Statt

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

kann man wegen Rundungsfehlern nur gestörte Werte

$$\tilde{y}_{n+1} = \tilde{y}_n + hf(t_n, \tilde{y}_n) + \delta_n$$

berechnen. δ_n bezeichnet man als lokalen Rundungsfehler. Wir nehmen im Folgenden an, dass

$$|\delta_n| \leq \delta \quad \text{für alle } n = 0, 1, 2, \dots$$

gilt, wobei δ abhängig von der Maschinengenauigkeit ist. Für den globalen Rundungsfehler gilt dann analog zu Teil (c) des Beweises von Satz 8.3

$$\begin{aligned} \|\tilde{y}_n - y_n\| &\leq \delta \sum_{j=0}^{n-1} (1 + hL)^j \\ &\leq \delta \frac{(1 + hL)^n - 1}{(1 + hL) - 1} \\ &\leq \frac{\delta}{h} \frac{e^{nhL} - 1}{L} \leq M \frac{\delta}{h}, \end{aligned}$$



Copyright (C) Marlis Hochbruck, Karlsruhe Institute of Technology, SS 2010 - WS 2011/12

Copyright (C) Marlis Hochbruck, Karlsruhe Institute of Technology, SS 2010 - WS 2011/12

Copyright (C) Marlis Hochbruck, Karlsruhe Institute of Technology, SS 2010 - WS 2011/12

Copyright (C) Marlis Hochbruck, Karlsruhe Institute of Technology, SS 2010 - WS 2011/12

Copyright (C) Marlis Hochbruck, Karlsruhe Institute of Technology, SS 2010 - WS 2011/12



Copyright (C) Marlis Hochbruck, Karlsruhe Institute of Technology, SS 2010 - WS 2011/12

8.5 Runge-Kutta-Verfahren

Wir betrachten wieder das Anfangswertproblem (8.1) auf dem Intervall $I = [t_0, T]$. Als Motivation für Runge-Kutta-Verfahren schreiben wir für die exakte Lösung y

$$y(t_1) = y_0 + \int_{t_0}^{t_1} y'(t) dt, \quad t_1 = t_0 + h.$$

Nun approximieren wir das Integral auf der rechten Seite durch eine Quadraturformel mit den Knoten c_1, \dots, c_s und den Gewichten b_1, \dots, b_s :

$$y(t_1) \approx y_0 + h \sum_{i=1}^s b_i y'(t_0 + c_i h).$$

Aus der Differentialgleichung wissen wir, dass

$$y'(t_0 + c_i h) = f(t_0 + c_i h, y(t_0 + c_i h))$$

gilt. Leider treten in der Quadraturformel die Werte der unbekannten Lösung an den Stellen $t_0 + c_i h$, $i = 1, \dots, s$, auf. Diese kann man nun erneut durch Quadraturformeln approximieren:

$$y(t_0 + c_i h) = y_0 + \int_{t_0}^{t_0 + c_i h} y'(t) dt \approx y_0 + h \sum_{j=1}^s a_{ij} y'(t_0 + c_j h).$$

Ein Runge-Kutta-Verfahren ist damit wie folgt definiert:

$$\begin{aligned} y_1 &= y_0 + h \sum_{i=1}^s b_i Y'_i, \\ Y'_i &= f(t_0 + c_i h, Y_i), \\ Y_i &= y_0 + h \sum_{j=1}^s a_{ij} Y'_j. \end{aligned} \tag{8.6}$$

Falls $a_{ij} = 0$ für $j \geq i$ gilt, können die Y_i nacheinander explizit berechnet werden, vgl. Algorithmus 8.1.

Algorithmus 8.1 Explizites Runge-Kutta-Verfahren

```

for  $i = 1$  to  $s$  do
     $Y_i = y_0 + h \sum_{j=1}^{i-1} a_{ij} Y'_j$ 
     $Y'_i = f(t_0 + c_i h, Y_i)$ 
end for
 $y_1 = y_0 + h \sum_{i=1}^s b_i Y'_i$ 

```

Im nächsten Schritt nimmt man y_1 als Startwert, berechnet y_2 , usw. Ein Runge-Kutta-Verfahren ist durch die Koeffizienten a_{ij} , b_j , c_j festgelegt. Üblicherweise stellt man es in einem Tableau dar:

$$\begin{array}{c|c} c_i & a_{ij} \\ \hline & b_j \end{array}$$

Beispiel. Einige Beispiele expliziter Runge-Kutta-Verfahren:

(a) Explizites Euler-Verfahren:

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

$Y_1 = y_0$, $Y'_1 = f(t_0, Y_1)$, $y_1 = y_0 + hY'_1$. Vom expliziten Euler-Verfahren wissen wir bereits, dass der lokale Fehler $O(h^2)$ ist.

(b) Verfahren von Runge:

$$y_1 = y_0 + hf\left(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}f(t_0, y_0)\right)$$

oder

$$\begin{array}{ll} Y_1 = y_0, & Y'_1 = f(t_0, y_0), \\ Y_2 = y_0 + \frac{h}{2}Y'_1 & Y'_2 = f\left(t_0 + \frac{h}{2}, Y_2\right), \\ y_1 = y_0 + hY'_2. \end{array}$$

In kompakter Schreibweise ergibt sich

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

Um den Fehler des Verfahrens von Runge zu berechnen, führen wir eine Taylor-Entwicklung nach Potenzen von h durch. Für die numerische Lösung y_1 gilt

$$y_1 = y_0 + hf(t_0, y_0) + \frac{h^2}{2} \left[\frac{\partial f}{\partial t}(t_0, y_0) + \frac{\partial f}{\partial y}(t_0, y_0)f(t_0, y_0) \right] + O(h^3),$$

für die exakte Lösung $y(t_1)$

$$\begin{aligned} y(t_0 + h) &= y_0 + hy'(t_0) + \frac{h^2}{2}y''(t_0) + O(h^3) \\ &= y_0 + hf(t_0, y_0) + \frac{h^2}{2} \left[\frac{\partial f}{\partial t}(t_0, y_0) + \frac{\partial f}{\partial y}(t_0, y_0)f(t_0, y_0) \right] + O(h^3), \end{aligned}$$

denn durch Differenzieren nach t erhalten wir aus $y' = f(t, y(t))$ für die zweite Ableitung

$$y'' = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}y' = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}f.$$

Der lokale Fehler des Verfahrens von Runge ist damit $y_1 - y(t_0 + h) = O(h^3)$.

(c) Klassisches Runge-Kutta-Verfahren (Kutta, 1901):

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

Das Verfahren hat einen lokalen Fehler von $O(h^5)$, wie wir später sehen werden.

Man erkennt bereits am einfachen Beispiel des Verfahrens von Runge, dass die Untersuchung des lokalen Fehlers ohne systematisches Vorgehen sehr aufwändig werden wird. Versuchen Sie einmal, die Aussage über den lokalen Fehler in Teil (c) “von Hand” nachzurechnen. \diamond

Definition 8.4. Ein Runge-Kutta-Verfahren hat **Ordnung** p genau dann, wenn für jedes Anfangswertproblem (8.1) mit $f \in C^{p+1}$ gilt

$$y_1 - y(t_0 + h) = O(h^{p+1}).$$

In den obigen Beispielen haben wir folgende Ordnungen:

Euler-Verfahren:	Ordnung $p = 1$
Verfahren von Runge:	Ordnung $p = 2$
Verfahren von Kutta:	Ordnung $p = 4$

Satz 8.5. Sei $f \in C^{p+1}$. Hat das Runge-Kutta-Verfahren die Ordnung p , dann gilt für den globalen Fehler

$$\|y_n - y(t_n)\| \leq Mh^p,$$

wobei die Konstante M unabhängig von n und h mit $t_n = t_0 + nh \in I = [t_0, T]$ ist.

Beweis. Es gilt

$$y_{n+1} = y_n + h\Phi(t_n, y_n, h), \quad \text{wobei} \quad \Phi(t, y, h) = \sum_{i=1}^s b_i Y'_i(t, y, h),$$

$$Y'_i(t, y, h) = f(t + c_i h, Y_i(t, y, h)), \quad Y_i(t, y, h) = y + h \sum_{j=1}^s a_{ij} Y'_j(t, y, h)$$

- (a) Lokaler Fehler: $\leq Ch^{p+1}$ nach Definition der Ordnung (und wegen der Kompaktheit).
- (b) Fehlerfortpflanzung: Die Funktion Φ ist (lokal) Lipschitz-stetig als Hintereinanderausführung (lokal) Lipschitz-stetiger Funktionen (ggf. mit dem Satz über implizite Funktionen). Somit folgt aus

$$v_{n+1} = v_n + h\Phi(t_n, v_n, h),$$

$$w_{n+1} = w_n + h\Phi(t_n, w_n, h)$$

die Abschätzung

$$\|v_{n+1} - w_{n+1}\| \leq (1 + hL_\Phi) \|v_n - w_n\|.$$

- (c) Wie im Beweis von Satz 8.3 können wir das Resultat mit $M = C(e^{L_\Phi(T-t_0)} - 1)/L_\Phi$ schließen.

□

8.6 Taylor-Entwicklung und Bäume

Wir betrachten das autonome Anfangswertproblem

$$y' = f(y), \quad y(t_0) = y_0,$$

wobei $f : U \rightarrow \mathbb{R}^d$ mit $U \subset \mathbb{R}^d$ als beliebig oft differenzierbar vorausgesetzt wird. Der Satz von Taylor besagt

$$y(t_0 + h) = \sum_{k=0}^p y^{(k)}(t_0) \frac{h^k}{k!} + O(h^{p+1}), \quad h \rightarrow 0.$$

Höhere Ableitungen der Lösung y erhält man durch wiederholtes Differenzieren der Differentialgleichung. Direkt aus der Differentialgleichung liest man

$$y'(t_0) = \underset{\bullet}{f(y_0)}$$

ab. Durch Differenzieren und Einsetzen der Differentialgleichung erhalten wir

$$y''(t_0) = \underset{\swarrow}{(f'f)(y_0)}$$

Nochmalige Differentiation ergibt

$$y'''(t_0) = \underset{\swarrow}{(f''(f, f))(y_0)} + \underset{\searrow}{(f'f'f)(y_0)}$$

sowie

$$\begin{aligned} y^{(4)}(t_0) = & \underset{\swarrow}{(f'''(f, f, f))(y_0)} + \underset{\searrow}{3(f''(f'f, f))(y_0)} \\ & + \underset{\swarrow}{(f'f''(f, f))(y_0)} + \underset{\searrow}{(f'f'f'f)(y_0)} \end{aligned}$$

Man erkennt, dass schon bei dreimaligem Ableiten die Terme sehr schnell kompliziert werden. Deshalb ist es nützlich, eine graphische Darstellung einzuführen. Es wird jeder Ableitung ein (bewurzelter) Baum zugeordnet. Wir versehen f mit einem Knoten und $f^{(m)}$ ebenfalls mit einem Knoten, von dem jedoch m Äste ausgehen. In obigen Formeln stehen die Bäume jeweils unter den entsprechenden Ausdrücken, den sogenannten elementaren Differentialen. Dies motiviert folgende rekursive Definition:

Definition 8.6. Sei \mathcal{T} die Menge der Bäume, die rekursiv definiert werden durch:

- (a) $\tau_{11} = \bullet \in \mathcal{T}$,

(b) Falls $\tau_1, \dots, \tau_m \in \mathcal{T}$, dann gilt $[\tau_1, \dots, \tau_m] \in \mathcal{T}$.

Hier ist $[\tau_1, \dots, \tau_m]$ (ein ungeordnetes m -Tupel) der Baum, den man erhält, wenn man die Wurzeln der Bäume τ_1, \dots, τ_m über m Äste mit einem neuen Knoten verbindet. Dieser ist die neue Wurzel des Baums $[\tau_1, \dots, \tau_m]$.

In Abbildung 8.4 sind einige Beispiele von Bäumen dargestellt.

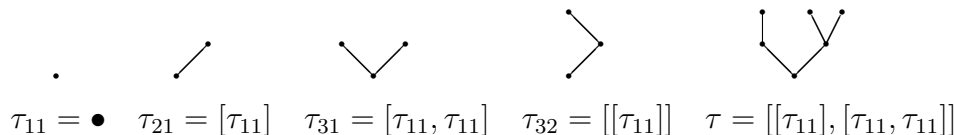


Abb. 8.4: Beispiele von Bäumen

Definition 8.7. Unter der Ordnung $\rho(\tau)$ eines Baums $\tau \in \mathcal{T}$ versteht man die Anzahl seiner Knoten.

Offensichtlich gilt

$$\begin{aligned} \rho(\bullet) &= 1, \\ \rho(\tau) &= 1 + \rho(\tau_1) + \dots + \rho(\tau_m), \quad \text{für } \tau = [\tau_1, \dots, \tau_m]. \end{aligned}$$

Zum Beispiel haben die Bäume, die zu $y^{(3)}$ gehören, Ordnung 3, und die zu $y^{(4)}$ Ordnung 4. In Tabelle 8.1 ist die Anzahl der Bäume bis zur Ordnung 10 dargestellt.

Tabelle 8.1: Die Anzahl der Bäume bis zur Ordnung 10

Ordnung r	1	2	3	4	5	6	7	8	9	10
Anz. d. Bäume	1	1	2	4	9	20	48	115	286	719

Den genauen Zusammenhang zwischen einem Baum und seinem elementarem Differential beschreibt

Definition 8.8. Für jeden Baum $\tau \in \mathcal{T}$ definieren wir eine Abbildung $F(\tau) : U \rightarrow U$, das **elementare Differential**, rekursiv durch:

- (a) $F(\bullet)(y) = F(\tau_{11})(y) = f(y)$,
- (b) $F(\tau)(y) = f^{(m)}(y)(F(\tau_1)(y), \dots, F(\tau_m)(y))$ für $\tau = [\tau_1, \dots, \tau_m]$.

Satz 8.9. Für die Lösung von $y' = f(y)$, $y(t_0) = y_0$ gilt

$$y^{(k)}(t_0) = \sum_{\substack{\tau \in \mathcal{T} \\ \rho(\tau) = k}} \alpha(\tau) F(\tau)(y_0), \quad k = 1, 2, 3, \dots$$

für gewisse Konstanten $\alpha(\tau)$, die unabhängig von der Differentialgleichung sind.

Beweis. Der Beweis erfolgt mit Induktion nach k . Für $k = 1, 2, 3, 4$ haben wir die Aussage bereits oben gesehen. Nehmen wir also an, die Behauptung wäre richtig für $k - 1$.

$y^{(k)}(t_0)$ ist eine Linearkombination von

$$f^{(m)}(y_0)\left(y^{(k_1)}(t_0), \dots, y^{(k_m)}(t_0)\right) \quad \text{mit } k_1 + \dots + k_m = k - 1, \quad 1 \leq m \leq k - 1.$$








Nach Induktionsvoraussetzung ist $y^{(l)}(t_0)$ für $l < k$ eine Linearkombination von elementaren Differentialen der Ordnung l . Daher kann man $y^{(k)}(t_0)$ als Linearkombination von

$$f^{(m)}(y_0)\left(F(\tau_1)(y_0), \dots, F(\tau_m)(y_0)\right) = F(\tau)(y_0)$$

mit $\rho(\tau_1) + \dots + \rho(\tau_m) = k - 1$ schreiben. Weil der Baum $\tau = [\tau_1, \dots, \tau_m]$ genau $\rho(\tau) = k$ Knoten hat, folgt die Behauptung. \square

Die Bäume mit den zugehörigen Differentialen finden Sie in Tabelle 8.2 auf Seite 186.

Tabelle 8.2: Bäume und die zugehörigen elementaren Differentialen

r	τ	Graph	$\gamma(\tau)$	$\alpha(\tau)$	$F(\tau)$	$\Phi_i(\tau)$
1	τ	\cdot	1	1	f	1
2	τ_{21}		2	1	$f'f$	$\sum_k a_{ik}$
3	τ_{31}		3	1	$f''(f, f)$	$\sum_{k,l} a_{ik} a_{il}$
	τ_{32}		6	1	$f'f'f$	$\sum_{k,l} a_{ik} a_{kl}$
4	τ_{41}		4	1	$f'''(f, f, f)$	$\sum_{k,l,m} a_{ik} a_{il} a_{im}$
	τ_{42}		8	3	$f''(f'f, f)$	$\sum_{k,l,m} a_{ik} a_{kl} a_{im}$
	τ_{43}		12	1	$f'f''(f, f)$	$\sum_{k,l,m} a_{ik} a_{kl} a_{km}$
	τ_{44}		24	1	$f'f'f'f$	$\sum_{k,l,m} a_{ik} a_{kl} a_{lm}$

8.7 Ordnungsbedingungen für Runge-Kutta-Verfahren

Die Ordnungsbedingungen von Runge-Kutta-Verfahren erhalten wir durch Vergleich der Taylor-Entwicklungen der numerischen Lösung y_1 und der exakten Lösung $y(t_0 + h)$.

Für die autonome Differentialgleichung (8.2) lautet das Runge-Kutta-Verfahren

$$y_1 = y_0 + h \sum_{i=1}^s b_i Y'_i, \quad Y'_i = f(Y_i) \quad (8.7)$$

$$Y_i = y_0 + h \sum_{j=1}^s a_{ij} Y'_j. \quad (8.8)$$

Wir fassen nun y_1 , Y_i und Y'_i als Funktionen der Schrittweite h auf. Dann gilt:

$$\begin{aligned} Y'_i &= f(Y_i) \\ \dot{Y}'_i &= f'(Y_i) \dot{Y}_i, \quad \cdot = \frac{d}{dh} \\ \ddot{Y}'_i &= f''(Y_i)(\dot{Y}_i, \dot{Y}_i) + f'(Y_i) \ddot{Y}_i. \end{aligned}$$

Allgemein ist $(Y'_i)^{(k-1)}$ eine Linearkombination von $f^{(m)}(Y_i^{(k_1)}, \dots, Y_i^{(k_m)})$ mit $k_1 + \dots + k_m = k - 1$, $1 \leq m \leq k - 1$ mit denselben Koeffizienten wie bei $y^{(k)}$.

Leiten wir (8.8) l -mal nach h ab, so ergibt sich mit der Leibnizregel

$$(uv)^{(l)} = \sum_{j=0}^l \binom{l}{j} u^{(j)} v^{(l-j)}$$

für $u = h$ und $v = Y'_j$ wegen $\binom{l}{1} = l$

$$Y_i^{(l)}(0) = l \sum_{j=1}^s a_{ij} (Y'_j)^{(l-1)}(0).$$

Einsetzen liefert, dass $(Y'_i)^{(k-1)}(0)$ eine Linearkombination von

$$k_1 \cdots k_m \sum_{j_1, \dots, j_m=1}^s a_{ij_1} \cdots a_{ij_m} f^{(m)}\left((Y'_{j_1})^{(k_1-1)}(0), \dots, (Y'_{j_m})^{(k_m-1)}(0)\right)$$

mit denselben Koeffizienten wie zuvor ist, wobei $k_1 + \dots + k_m = k - 1$, $1 \leq m \leq k - 1$.

Die Idee ist nun, diese Formel rekursiv zu verwenden.

Definition 8.10. Für $\tau \in \mathcal{T}$ setzen wir rekursiv

$$\begin{aligned} \varphi'_i(\bullet) &= 1, \\ \varphi'_i(\tau) &= \rho(\tau_1) \cdots \rho(\tau_m) \sum_{j_1, \dots, j_m} a_{ij_1} \cdots a_{ij_m} \varphi'_{j_1}(\tau_1) \cdots \varphi'_{j_m}(\tau_m), \quad \tau = [\tau_1, \dots, \tau_m], \\ \varphi(\tau) &= \rho(\tau) \sum_{i=1}^s b_i \varphi'_i(\tau). \end{aligned}$$

Satz 8.11. Für die numerische Lösung gilt

$$y_1^{(k)}(0) = \sum_{\substack{\tau \in \mathcal{T} \\ \rho(\tau)=k}} \varphi(\tau) \alpha(\tau) F(\tau)(y_0), \quad k = 1, 2, 3, \dots$$

mit denselben Koeffizienten $\alpha(\tau)$ wie in Satz 8.9.

$$\gamma(t) = \begin{array}{c} \text{Diagram 1} \\ \text{Diagram 2} \\ \text{Diagram 3} \end{array} = 6 \cdot 2 \cdot 3 \cdot 1 \cdot 1 \cdot 1 = 36$$

Abb. 8.5: Die Berechnung der Koeffizienten $\gamma(t)$

Beweis. Obige Formel und Induktion liefern

$$(Y'_i)^{(k-1)}(0) = \sum_{\substack{\tau \in \mathcal{T} \\ \rho(\tau)=k}} \varphi'_i(\tau) \alpha(\tau) F(\tau)(y_0)$$

und außerdem

$$y_1^{(k)}(0) = k \sum_{i=1}^s b_i (Y'_i)^{(k-1)}(0).$$

□

Satz 8.12. Das Runge-Kutta-Verfahren (8.6) hat Ordnung p genau dann, wenn

$$\varphi(\tau) = 1 \quad \text{für alle Bäume } \tau \in \mathcal{T} \text{ mit } \rho(\tau) \leq p. \quad (8.9)$$

Beweis. Nach Satz 8.9 und 8.11 gilt für den lokalen Fehler des Runge-Kutta-Verfahrens

$$y(t_0 + h) - y_1 = \sum_{\substack{\tau \in \mathcal{T} \\ \rho(\tau) \leq p}} (1 - \varphi(\tau)) \alpha(\tau) F(\tau)(y_0) \frac{h^{\rho(\tau)}}{\rho(\tau)!} + O(h^{p+1}).$$

(8.9) ist also hinreichend dafür, dass das Verfahren die Ordnung p hat. Die Notwendigkeit der Bedingungen folgt aus der Unabhängigkeit der elementaren Differentiale. □

Seien $\Phi_i(\tau)$ bzw. $\Phi(\tau)$ wie φ'_i bzw. φ definiert, jedoch ohne die Faktoren $\rho(\tau_j)$ bzw. $\rho(\tau)$. Dann ist

$$\varphi(\tau) = \gamma(\tau) \Phi(\tau),$$

wobei $\gamma(\tau)$ rekursiv definiert ist durch

$$\begin{aligned} \gamma(\bullet) &= 1, \\ \gamma(\tau) &= \rho(\tau) \gamma(\tau_1) \cdots \gamma(\tau_m), \quad \text{für } \tau = [\tau_1, \dots, \tau_m]. \end{aligned}$$

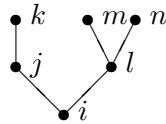
Die praktische Berechnung der Koeffizienten $\gamma(t)$ zeigt Abbildung 8.5.

Die Bedingungsgleichung in Satz 8.12 ist daher äquivalent zu

$$\sum_{i=1}^s b_i \Phi_i(\tau) = \Phi(\tau) = \frac{1}{\gamma(\tau)}. \quad (8.10)$$

Die Ordnungsbedingungen können direkt von den Bäumen abgelesen werden. Man gibt jedem Knoten eines Baums $\tau \in T$ einen Buchstaben i, j, k, l, \dots , wobei die Wurzel den Index i bekommt. Die linke Seite von (8.10) ist dann die Summe über i, j, k, l, \dots , wobei man die Summanden als Produkt von b_i mit Faktoren erhält, in denen a_{lm} genau dann auftritt, wenn m Sohn von l ist.

Beispiel. Für den Baum



lautet die Ordnungsbedingung

$$\sum_{i,j,k,l,m,n} b_i a_{ij} a_{jk} a_{il} a_{lm} a_{ln} = \frac{1}{6 \cdot 2 \cdot 3}.$$

Weitere Beispiele sind Tabelle 8.2 zu entnehmen. \diamond

8.8 Konstruktion expliziter Runge-Kutta-Verfahren

Bei einem Runge-Kutta-Verfahren (8.6) benötigt man für s Stufen genau s Auswertungen von f . Diese sind in der Regel teuer. Also möchte man Runge-Kutta-Verfahren konstruieren, bei denen s klein, die Ordnung p jedoch groß ist.

Satz 8.13. *Ein explizites, s -stufiges Runge-Kutta-Verfahren hat höchstens Ordnung s .*

Beweis. Die Bedingungsgleichung für den längsten Baum der Ordnung p lautet:

$$\sum_{j_1, \dots, j_p=1}^s b_{j_1} a_{j_1, j_2} \cdots a_{j_{p-1}, j_p} = \frac{1}{p!}.$$

Bei expliziten Runge-Kutta-Verfahren ist $a_{ij} = 0$ für $j \geq i$. Damit ein Summand von Null verschieden ist, muss

$$1 \leq j_p < j_{p-1} < \cdots < j_1 \leq s$$

sein, also $p \leq s$. \square

Wir konstruieren jetzt explizite Runge-Kutta-Verfahren mit maximaler Ordnung $p = s$. Da es genügt, autonome Differentialgleichungen zu betrachten (Übung) können wir die vereinfachte Annahme

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

treffen.

$p = 1$: Die einzige Ordnungsbedingung ist $b_1 = 1$, was wegen $c_1 = 0$ gerade das explizite Euler-Verfahren liefert.

$p = 2$: Tabelle 8.2 liefert für den Baum τ_{11} die Ordnungsbedingung $b_1 + b_2 = 1$ und für den Baum τ_{21} die Gleichung $b_1 c_1 + b_2 c_2 = \frac{1}{2}$. Wegen $c_1 = 0$ ergibt sich das Tableau

$$\begin{array}{c|cc} 0 & 0 & \\ c_2 & c_2 & 0 \\ \hline & 1 - \frac{1}{2c_2} & \frac{1}{2c_2} \end{array}.$$

Für $c_2 = \frac{1}{2}$ ist das das Verfahren von Runge (1895) und für $c_2 = 1$ das Verfahren von Heun (1900).

$p = 3$: Allgemeine Lösung als Übung.

$p = 4$: Zum Beispiel das “klassische” Runge-Kutta-Verfahren.

$p = 5$: Es existiert kein 5-stufiges Runge-Kutta-Verfahren der Ordnung 5.

Schließlich wollen wir noch eine Übersicht über erreichbare Ordnungen geben:

Ordnung p	1	2	3	4	5	6	7	8	9	10
minimales s	1	2	3	4	6	7	9	11	≥ 12	≥ 13
# Bedgl.	1	2	4	8	17	37	85	200	486	1205

8.9 Schrittweitensteuerung durch eingebettete Runge-Kutta-Verfahren

Ähnlich wie bei den zusammengesetzten Quadraturformeln in Kapitel 1 ist es problematisch, in jedem Zeitschritt dieselbe Schrittweite h zu verwenden. Im Allgemeinen wird nämlich der lokale Fehler bei einigen Schritten sehr klein, bei anderen jedoch groß sein. Es ist daher ein hoher Rechenaufwand nötig, um die gewünschte Genauigkeit zu erreichen.

Das Ziel einer Schrittweitensteuerung ist es, das Intervall $[t_0, T]$ so zu unterteilen, dass die *lokalen* Fehler in jedem Teilintervall ungefähr gleich groß und kleiner als eine vorgegebene Fehlerschranke (“Toleranz”) sind. Dazu benötigen wir wieder eine brauchbare numerische Fehlerschätzung.

Die Idee haben wir schon in Abschnitt 1.9 kennengelernt. Wir verwenden zwei Runge-Kutta-Verfahren mit denselben Funktionsauswertungen (eingebettete Verfahren).

$$\begin{aligned}
 Y'_i &= f(t_0 + c_i h, Y_i) & Y_i &= y_0 + h \sum_{j=1}^{i-1} a_{ij} Y'_j, & i &= 1, \dots, s, \\
 y_1 &= y_0 + h \sum_{i=1}^s b_i Y'_i, & \hat{y}_1 &= y_0 + h \sum_{i=1}^s \hat{b}_i Y'_i.
 \end{aligned}$$

Die beiden eingebetteten Verfahren lassen sich im Tableau übersichtlich darstellen:

$$\begin{array}{c|c}
 c_i & a_{ij} \\
 \hline
 & b_j \\
 \hline
 & \hat{b}_j
 \end{array}$$

Dabei habe y_1 die Ordnung p und \hat{y}_1 die Ordnung $p - 1$. Für die lokalen Fehler gilt dann

$$\begin{aligned}
 y_1 - y(t_0 + h) &= O(h^{p+1}), \\
 \hat{y}_1 - y(t_0 + h) &= Ch^p + O(h^{p+1}), & C &= C(t_0, y_0).
 \end{aligned}$$

Der lokale Fehler von \hat{y}_1 kann durch

$$\text{err} := y_1 - \hat{y}_1 = h \sum_{i=1}^s e_i Y'_i, \quad e_i = b_i - \hat{b}_i \quad (8.11)$$

bis auf $O(h^{p+1})$ geschätzt werden. Die Bedingungsgleichungen für die beiden Verfahren lauten nach Satz 8.12

$$\begin{aligned} \sum_{i=1}^s b_i \Phi_i(\tau) &= \frac{1}{\gamma(\tau)} && \text{für } \rho(\tau) \leq p, \\ \sum_{i=1}^s \hat{b}_i \Phi_i(\tau) &= \frac{1}{\gamma(\tau)} && \text{für } \rho(\tau) \leq p-1. \end{aligned}$$

Damit $y_1 - \hat{y}_1$ eine sinnvolle Fehlerschätzung ist, darf \hat{y}_1 keine Approximation der Ordnung p sein:

$$\sum_{i=1}^s \hat{b}_i \Phi_i(\tau) \neq \frac{1}{\gamma(\tau)} \quad \text{für } \rho(\tau) = p.$$

Auf der Schätzung **err** des lokalen Fehlers beruht die folgende Schrittweitensteuerung. Man bestimmt die neue Schrittweite h so, dass komponentenweise

$$|y_1 - \hat{y}_1| \leq sc, \quad sc^I = Atol^I + \max\{|y_0^I|, |y_1^I|\} \cdot Rtol^I,$$

wobei $Atol^I$ und $Rtol^I$ vom Benutzer vorgegebene Toleranzen sind und mit dem oberen Index I die I -te Komponente des Vektors bezeichnet wird. Für $Atol^I = 0$, $I = 1, \dots, d$, misst man relative Fehler, für $Rtol^I = 0$, $I = 1, \dots, d$, absolute Fehler. In der Praxis verwendet man häufig eine gemischte Skalierung, bei denen beide Toleranzen positiv sind. Als Maß für den Fehler setzt man

$$\|\mathbf{err}\| = \sqrt{\frac{1}{d} \sum_{J=1}^d \left(\frac{\mathbf{err}^J}{sc^J} \right)^2}. \quad (8.12)$$

Hat man für ein h die Werte y_1 , \hat{y}_1 und $\|\mathbf{err}\|$ berechnet, so ist $\|\mathbf{err}\| \approx Ch^p$. Das ideale h_{opt} wäre jenes mit $\|Ch_{opt}^p\| = 1$, d. h.

$$\|\mathbf{err}\| \left(\frac{h_{opt}}{h} \right)^p = 1 \quad \Longleftrightarrow \quad h_{opt} = h \sqrt[p]{\frac{1}{\|\mathbf{err}\|}}.$$

Ist $\|\mathbf{err}\| > 1$, dann wird der aktuelle Schritt verworfen und mit der Schrittweite h_{opt} erneut gerechnet. Für $\|\mathbf{err}\| \leq 1$ ist es wegen $C(t_0 + h, y_1) = C(t_0, y_0) + O(h)$ sinnvoll, dieses h_{opt} im nächsten Schritt zu wählen. Als Anfangswert für den nächsten Schritt wählt man den genaueren Wert y_1 .

In einem guten Code sollte man vorsichtig mit derartigen Schätzungen umgehen. Daher führt man einige "Sicherheitsfaktoren" ein. Zum Beispiel multipliziert man die optimale Schrittweite noch mit einem Faktor fac , etwa $fac = 0.8, 0.9, 0.25^{1/p}$, damit der nächste Zeitschritt mit hoher Wahrscheinlichkeit akzeptiert wird. Zudem verbietet man, dass die Schrittweite zu schnell erhöht oder reduziert wird und setzt schließlich

$$h_{neu} = h \cdot \min\{facmax, \max\{facmin, fac \cdot \frac{1}{\|\mathbf{err}\|^{1/p}}\}\}.$$

Mögliche Werte sind zum Beispiel $facmin = 1/3$, $facmax$ zwischen 1.5 und 5. Nach einem verworfenen Schritt sollte man $facmax = 1$ setzen, also eine unmittelbare Erhöhung verbieten.

In Algorithmus 8.2 ist eine mögliche Implementierung der Schrittweitensteuerung angegeben.

Algorithmus 8.2 Adaptives Runge-Kutta-Verfahren

```

function y=rkv(fcn, t, y, T, Atol, Rtol, h),
    {fcn rechte Seite der Dgl., (t, y) Anfangswerte,
    T Ende des Zeitintervalls, h Anfangsschrittweite}
    facmin = 1/3, facmax = 3, fac = 0.9
    while t < T do
        h = min(h, T - t), fertig = false
        while not fertig do
            berechne err
            fertig = (||err|| < 1)
            if fertig then {akzeptiere Schritt}
                berechne y1, setze y = y1, t = t + h
            end if
            {Schrittweite für nächsten Schritt, p = Ordnung:}
            h = h · min{facmax, max{facmin, fac ·  $\frac{1}{\|err\|^{1/p}}$ }}
        end while
    end while

```

Beispiel 8.1. Wir betrachten die van der Pol-Gleichung

$$y'' + \mu(y^2 - 1)y' + y = 0,$$

welche als System erster Ordnung die Form

$$\begin{aligned} y_1' &= y_2, & y_1(0) &= 2 \\ y_2' &= \mu(1 - y_1^2)y_2 - y_1, & y_2(0) &= 0 \end{aligned}$$

hat. In Abbildung 8.6 wurde dieses System für $\mu = 2$ mit einem Runge-Kutta-Verfahren fünfter Ordnung (`radau5`) einmal mit und einmal ohne Schrittweitensteuerung gelöst. In rot ist für jeden Zeitschritt die Schätzung (8.11) des lokalen Fehlers über der Zeit aufgetragen. Man erkennt, dass bei konstanter Schrittweite der lokale Fehler über mehrere Zehnerpotenzen variiert und in den beiden Phasen, in denen die Lösung sich stärker ändert, über der erlaubten Toleranz liegt. Die Schrittweitensteuerung führt dazu, dass die lokalen Fehlerschätzungen weit weniger variieren, und zwar dadurch, dass in den beiden oben genannten Phasen kleinere Zeitschrittweiten verwendet und auch einige wenige Schritte wiederholt werden. In den glatten Phasen wird die Schrittweite anschließend wieder erhöht. \diamond

8.10 Stetige numerische Lösung bei Runge-Kutta-Verfahren

Nach einem Zeitschritt mit der Schrittweite h ausgehend von y_0 liefert ein Runge-Kutta-Verfahren nur eine Näherung für die Lösung $y(t_0 + h)$ nicht jedoch für $y(t_0 + \theta h)$ für beliebiges $0 < \theta < 1$. Die Zwischenstellen t_j werden im Allgemeinen durch eine Schrittweitensteuerung definiert. In manchen Anwendungen ist es aber nötig, die Lösung auch an beliebigen Zwischenstellen zu berechnen. Dies sollte natürlich ohne großen Mehraufwand, insbesondere ohne weitere Auswertungen der Funktion f geschehen.

Im Runge-Kutta-Verfahren berechnen wir

$$y_1 = y_0 + h \sum_{i=1}^s b_i Y_i', \quad Y_i' = f(t_0 + c_i h, Y_i), \quad Y_i = y_0 + h \sum_j a_{ij} Y_j'.$$

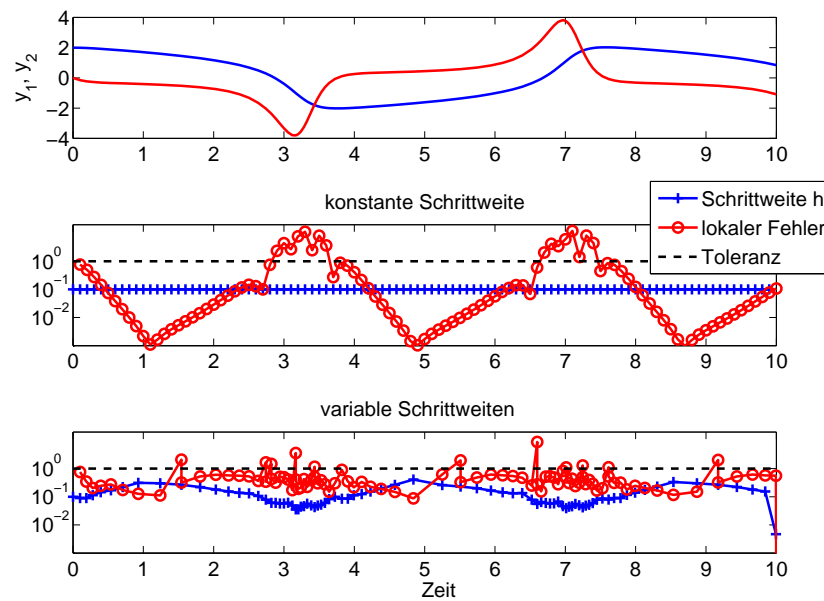


Abb. 8.6: Schrittweiten und lokale Fehler für die van der Pol-Gleichung. Im oberen Bild ist Lösung eingezeichnet (y_1 rot, y_2 blau).

Da wir keine neuen f -Auswertungen berechnen möchten, müssen Y_i und Y'_i aus dem Runge-Kutta-Verfahren verwendet werden. Die Idee besteht jetzt darin, variable Koeffizienten $b_i(\theta)$ einzusetzen und damit

$$y_\theta = y_0 + h \sum_{i=1}^{s^*} b_i(\theta) Y'_i \quad (8.13)$$

zu berechnen. In der Regel wird $s^* \geq s+1$ sein, da wir zumindest die erste Funktionsauswertung für den nächsten Schritt, $f(t_0+h, y_1)$, noch ohne Mehraufwand in die Formel einbeziehen können, wenn wir $a_{s+1,j} = b_j$, $j = 1, \dots, s+1$ setzen.

Die Koeffizienten $b_i(\theta)$ sollten so konstruiert werden, dass

$$y_\theta - y(t_0 + \theta h) = O(h^{p^*+1})$$

gilt, der lokale Fehler an den Zwischenstellen also die Ordnung p^* hat.

Satz 8.14. Die Approximation (8.13) hat die Ordnung p^* genau dann, wenn gilt

$$\sum_{i=1}^{s^*} b_i(\theta) \Phi_i(\tau) = \frac{\theta^{\rho(\tau)}}{\gamma(\tau)}, \quad \text{für } \rho(\tau) \leq p^*.$$

Beweis. Die k -te Ableitung der numerischen Lösung y_θ ist wie in Satz 8.11 gegeben, wenn wir nur b_i durch $b_i(\theta)$ ersetzen. Die k -te Ableitung der exakten Lösung $y(t_0 + \theta h)$ ist $\theta^k y^{(k)}(t_0)$. Die Behauptung folgt dann wie in Satz 8.12. \square

Schreiben wir die Polynome $b_i(\theta)$ als

$$b_i(\theta) = \sum_{j=1}^{p^*} b_{ij} \theta^j,$$

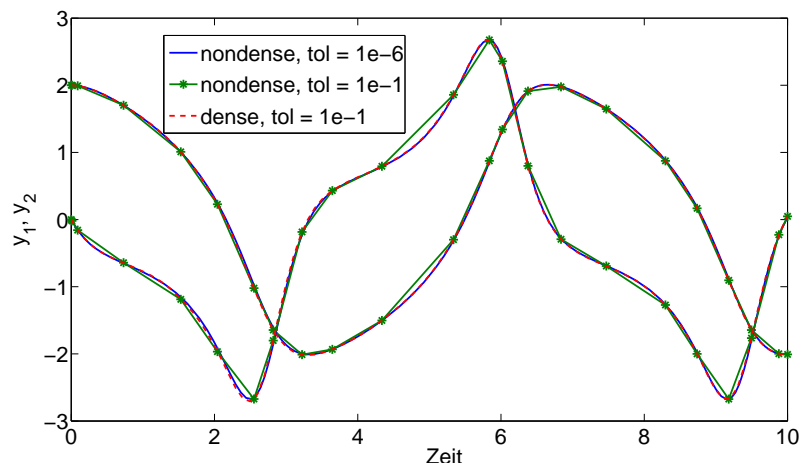


Abb. 8.7: Stetige numerische Lösung für die van der Pol-Gleichung

so führt Satz 8.14 auf ein lineares Gleichungssystem

$$\begin{pmatrix} 1 & \cdots & 1 \\ \Phi_1(\tau_{21}) & \cdots & \Phi_{s^*}(\tau_{21}) \\ \Phi_1(\tau_{31}) & \cdots & \Phi_{s^*}(\tau_{31}) \\ \vdots & & \vdots \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & \cdots \\ b_{21} & b_{22} & \cdots \\ \vdots & \vdots & \\ b_{s^*1} & b_{s^*2} & \cdots \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots \\ 0 & \frac{1}{\gamma(\tau_{21})} & 0 & \cdots \\ 0 & 0 & \frac{1}{\gamma(\tau_{31})} & \cdots \\ 0 & 0 & \frac{1}{\gamma(\tau_{32})} & \cdots \\ \vdots & \vdots & \vdots & \end{pmatrix}.$$

Es ist möglich, dass die Ordnung p^* der stetigen Runge-Kutta-Lösung kleiner als die des Verfahrens selbst ist. Im Allgemeinen ist die stetige numerische Lösung nicht auf dem gesamten Zeitintervall differenzierbar.

Eine einfachere Alternative, bei der man ohne Lösung eines linearen Gleichungssystems auskommt, ist die Verwendung von Hermite-Interpolationspolynomen. Unabhängig vom Verfahren stehen ja die Funktionswerte y_0 und y_1 sowie die Ableitungen $f_0 = f(t_0, y_0)$ und $f_1 = f(t_1, y_1)$ zur Verfügung. Das kubische Interpolationspolynom lautet dann

$$y_\theta = (1 - \theta)y_0 + \theta y_1 + \theta(\theta - 1)((1 - 2\theta)(y_1 - y_0) + (\theta - 1)hf_0 + \theta hf_1).$$

Man kann zeigen, dass Hermite-Interpolation ein Spezialfall von obigem Ansatz ist (Übung). Hat das Runge-Kutta-Verfahren Ordnung $p \geq 3$, so hat das stetige Runge-Kutta-Verfahren die Ordnung $p^* = 3$. Da Funktionswerte und erste Ableitungen auf der linken und rechten Seite zweier aufeinanderfolgender Zeitintervalle übereinstimmen, ist die Lösung auf dem gesamten Intervall $[t_0, T]$ stetig differenzierbar.

Beispiel. Wir betrachten noch einmal Beispiel 8.1. In Abbildung 8.7 ist die stetige numerische Lösung (rot) im Vergleich zu einer mit sehr viel höherer Genauigkeit berechneten Lösung (blau) dargestellt. Die vom Verfahren gewählten Schritte sind durch die grünen Sterne zu erkennen. In grün ist zusätzlich der interpolierende lineare Spline eingezeichnet. \diamond