Numerische Grundlagen der digitalen Simulation dynamischer Systeme

Heinrich Voss
Technische Universität Hamburg–Harburg
Arbeitsbereich Mathematik
voss @ tu-harburg.de

1 Einleitung

Dynamische Systeme sind mathematische Modelle zur Beschreibung der zeitlichen Entwicklung physikalischer, technischer, ökonomischer oder anderer Systeme. Diese Modelle können Differenzengleichungen oder gewöhnliche oder partielle Differentialgleichungen sein. Wir betrachten hier nur gewöhnliche Anfangswertaufgaben.

Nur in sehr seltenen Fällen können die Differentialgleichungen exakt gelöst werden. Meistens ist man auf numerische Verfahren angewiesen, um Näherungslösungen zu bestimmen. In diesem Versuch sollen Vertreter der wichtigsten Klassen dieser Verfahren ausprobiert werden, und es soll damit ein Gefühl für ihre Eigenschaften entwickelt werden. Dabei orientieren wir uns an den in MATLAB implementierten Methoden. Eine ausführlichere Darstellung des hier behandelten Stoffes finden Sie in dem Skript über Numerische Behandlung von Differentialgleichungen [7].

2 Einschrittverfahren

Wir betrachten die Anfangswertaufgabe

$$y' = f(x, y), \quad y(a) = y_0, \tag{1}$$

wobei die Lösung y im Intervall $[a, b]$ gesucht ist. Dabei kann y auch vektorwertig, also (1) ein Differentialgleichungssystem erster Ordnung sein.

Es sei $a = x_0 < x_1 < x_2 < \ldots < x_N =: b$ eine (nicht notwendig äquidistante) Zerlegung von $[a, b]$. Wir nehmen diese Zerlegung zunächst als gegeben an. Tatsächlich wird die Folge der x_j im Verfahren mitbestimmt und an das Verhalten der Lösung der Anfangswertaufgabe angepasst.

Da $f(x_n, y(x_n))$ gerade die Steigung $y'(x_n)$ der gesuchten Lösung $y(x)$ von (1) ist, gilt näherungsweise bei nicht zu großer Schrittweite $h_n := x_{n+1} - x_n$

$$\frac{1}{h_n}\left(y(x_{n+1}) - y(x_n)\right) \approx f(x_n, y(x_n)),$$

d.h.

$$y(x_{n+1}) = y(x_n) + h_n\, f(x_n, y(x_n)) + \varepsilon_n. \tag{2}$$

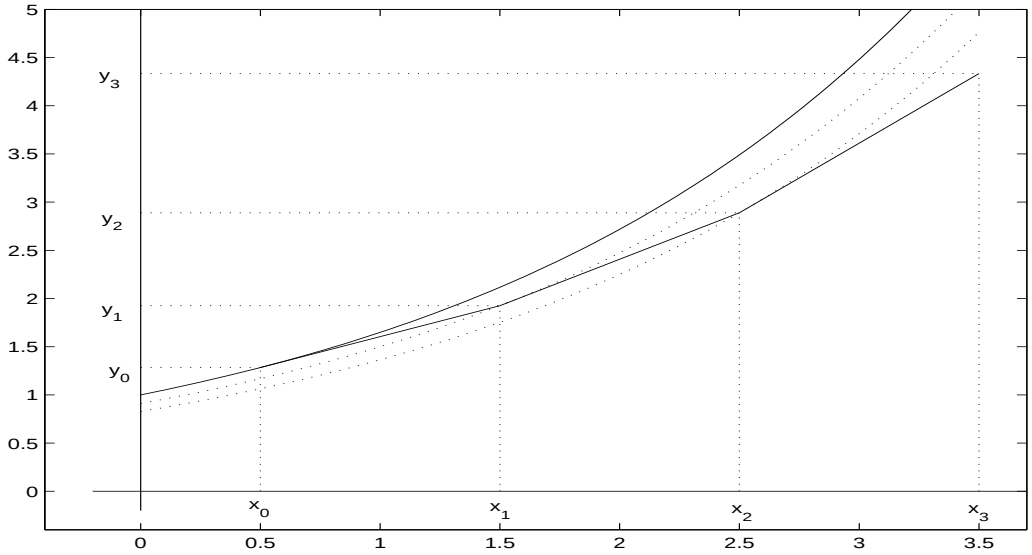

Abbildung 1: Knotenpunkte und Näherungswerte

Wir vernachlässigen nun in (2) den Fehler ε_n. Dann wird die entstehende Gleichung nicht mehr durch die Lösung $y(x_n)$ von (1) an den Knoten x_n erfüllt, sondern nur noch durch Näherungswerte y_n für $y(x_n)$. Wir bestimmen also die y_n ausgehend von y_0 durch das Verfahren

$$y_{n+1} = y_n + h_n f(x_n, y_n), \quad n = 0, 1, \ldots, N-1, \tag{3}$$

wobei $h_n := x_{n+1} - x_n$ ist.

Definition 1 *Das durch (3) beschriebene Verfahren zur approximativen Lösung der Anfangswertaufgabe (1) heißt das* **Eulersche Polygonzugverfahren**. *Es wurde 1768 von L. Euler beschrieben.*

Aufgabe 1 Wenden Sie auf die Anfangswertaufgabe

$$y' = y^2, \; y(0.8) = \frac{5}{6}, \quad x \in [0.8, 1.8]$$

mit der Lösung $y(x) = \dfrac{1}{2-x}$ das Polygonzugverfahren mit den äquidistanten Schrittweiten $h = \frac{1}{20}, \frac{1}{40}, \frac{1}{80}, \frac{1}{160}, \frac{1}{320}$ und $\frac{1}{640}$ an.
Wie verhält sich der Fehler bei Änderung der Schrittweite? □

Wir schätzen nun im allgemeinen Fall den Fehler ab. Dieser setzt sich aus zwei Anteilen zusammen: Wir haben im n-ten Schritt die Lösung $y(x_{n+1}; x_n, y(x_n))$ der Differentialgleichung $y' = f(x, y)$ mit dem Anfangswert $y(x_n)$ an der Stelle x_n zu bestimmen. Statt dessen betrachten wir die Anfangswertaufgabe

$$y' = f(x, y), \; y(x_n) = y_n$$

mit dem "falschen" Anfangswert y_n, und wir lösen diese auch nur näherungsweise, indem wir den Abbruchfehler vernachlässigen. Tatsächlich werden bei der Realisierung des

Verfahrens auf einem Rechner noch bei der Auswertung von $f(x_n, y_n)$ und den Rechenoperationen Rundungsfehler gemacht. Diese wollen wir aber bei den folgenden Betrachtungen außer Acht lassen.

Wir schreiben das Polygonzugverfahren in der Form

$$y_{n+1} - y_n - h_n\, f(x_n, y_n) = 0. \tag{4}$$

Setzt man hier an Stelle der Werte y_n die Werte $y(x_n)$ der Lösung von (1) an den Knoten x_n ein, so erhält man (vgl. (2))

$$y(x_{n+1}) - y(x_n) - h_n\, f(x_n, y(x_n)) =: \varepsilon(x_n, h_n). \tag{5}$$

Definition 2 *$\varepsilon(x_n, h_n)$ heißt der* **lokale Fehler** *(auch* **Abbruchfehler***) des Polygonzugverfahrens an der Stelle x_n bei der Schrittweite h_n.*

Subtrahiert man die Gleichung (4) von (5), so folgt

$$y(x_{n+1}) - y_{n+1} = y(x_n) - y_n + h_n\,(f(x_n, y(x_n)) - f(x_n, y_n)) + \varepsilon(x_n, h_n). \tag{6}$$

Definition 3

$$\delta_{n+1} := |y(x_{x+1}) - y_{n+1}|$$

heißt der **Fehler** *oder (zur besseren Unterscheidung)* **globale Fehler** *des Polygonzugverfahrens an der Stelle x_{n+1}.*

Erfüllt f auf $[a, b] \times \mathbb{R}$ eine Lipschitz Bedingung bzgl. y

$$|f(x, y) - f(x, z)| \le L|y - z| \qquad \text{für alle } y, z \in \mathbb{R} \text{ und alle } x \in [a, b],$$

so folgt aus (6) mit der Dreiecksungleichung und $\varepsilon_n := \varepsilon(x_n, h_n)$

$$\delta_{n+1} \le (1 + Lh_n)\,\delta_n + |\varepsilon_n|, \tag{7}$$

und durch vollständige Induktion erhält man hieraus

$$\delta_n \le e^{(b-a)L} \sum_{j=0}^{N-1} |\varepsilon_j|. \tag{8}$$

Bis auf eine multiplikative Konstante lässt sich der globale Fehler also durch die Summe der lokalen Fehler abschätzen.

Wir setzen nun weiter voraus, dass die Lösung y von (1) zweimal stetig differenzierbar ist. Dies ist z.B. erfüllt, wenn die rechte Seite f stetig differenzierbar in einer offenen Menge ist, die $\{(x, y(x)) : a \le x \le b\}$ enthält. Dann gilt nach dem Taylorschen Satz für den lokalen Fehler

$$\varepsilon_j = y(x_j + h_j) - y(x_j) - h_j y'(x_j) = \frac{1}{2} h_j^2 y''(x_j + \theta_j h_j)$$

mit einem $\theta_j \in (0, 1)$, und daher folgt für den globalen Fehler

$$\begin{aligned}
\delta_n &\le \frac{1}{2} e^{(b-a)L} \sum_{j=0}^{N-1} h_j^2 |y''(x_j + \theta_j h_j)| \le \frac{1}{2}\, e^{(b-a)L} \max_{a \le x \le b} |y''(x)| \sum_{j=0}^{N-1} h_j^2 \\
&\le \frac{1}{2}\, e^{(b-a)L} \max_{a \le x \le b} |y''(x)| \max_{j=0,\ldots,N-1} h_j \sum_{j=0}^{N-1} h_j =: C \cdot \max_{j=0,\ldots,N-1} h_j
\end{aligned}$$

mit einer von den gewählten Schrittweiten h_n unabhängigen Konstante C.

Hieraus liest man ab, dass bei Halbierung der Schrittweiten der Fehler ebenfalls halbiert wird. Man liest ferner ab, dass man $O(\delta^{-1})$ Schritte des Polygonzugverfahrens aufwenden muss, um den globalen Fehler δ zu erreichen (für den Fehler $\delta = 10^{-6}$ also $c \cdot 10^6$ Schritte).

In der Praxis wird man nicht mit vorgegebenen Schrittweiten rechnen, sondern die Schrittweite dem Lösungsverhalten anpassen. Dabei schätzt man wie bei den adaptiven Quadraturformeln den lokalen Fehler mit Hilfe einer zweiten Formel.

Wir verwenden hierzu zwei Schritte des Polygonzugverfahrens mit halber Schrittweite:

$$\begin{aligned} \tilde{y}_{n+\frac{1}{2}} &= y_n + \frac{h_n}{2} f(x_n, y_n) \\ \tilde{y}_{n+1} &= \tilde{y}_{n+\frac{1}{2}} + \frac{h_n}{2} f(x_n + \frac{h_n}{2}, \tilde{y}_{n+\frac{1}{2}}) \\ &= y_n + \frac{h_n}{2} f(x_n, y_n) + \frac{h_n}{2} f(x_n + \frac{h_n}{2}, y_n + \frac{h_n}{2} f(x_n, y_n)). \end{aligned}$$

Bezeichnet $z(x)$ die Lösung der Anfangswertaufgabe $y' = f(x, y)$, $y(x_n) = y_n$, und gilt $z \in C^3[a, b]$, so erhält man mit dem Taylorschen Satz für den lokalen Fehler des Polygonzugverfahrens

$$\varepsilon(x_n, h_n) := z(x_n + h_n) - (y_n + h_n\, f(x_n, y_n)) = \frac{1}{2}\, h_n^2 z''(x_n) + O(h_n^3) \tag{9}$$

und für den lokalen Fehler der zusammengesetzten Formel

$$\tilde{\varepsilon}(x_n, h_n) = z(x_n + h_n) - \tilde{y}_{n+1} = \frac{1}{4}\, h_n^2 z''(x_n) + O(h_n^3). \tag{10}$$

Durch Subtraktion dieser beiden Formeln erhält man

$$\tilde{y}_{n+1} - y_{n+1} = \frac{1}{4}\, h_n^2 z''(x_n) + O(h_n^3).$$

Vernachlässigt man den $O(h_n^3)$-Terms, so liefert (9) die Schätzung für den lokalen Fehler

$$\varepsilon(x_n, h_n) \approx \phi(x_n, h_n) := 2(\tilde{y}_{n+1} - y_{n+1}). \tag{11}$$

Zugleich erhält man mit

$$\hat{y}_{n+1} := 2\tilde{y}_{n+1} - y_{n+1} = y_n + h_n f(x_n + \frac{1}{2}h_n, y_n + \frac{1}{2} f(x_n, y_n)) \tag{12}$$

eine Näherung für $y(x_n + h_n)$ mit dem lokalen Fehler

$$\hat{\varepsilon}(x_n, h_n) = 2\,\tilde{\varepsilon}(x_n, h_n) - \varepsilon(x_n, h_n) = O(h_n^3).$$

Verfahren mit dieser Eigenschaft werden wir Verfahren der Ordnung 2 nennen (der lokale Fehler besitzt die Ordnung 3, aber beim Übergang zum globalen Fehler verliert man wie beim Euler Verfahren eine h-Potenz).

Die Formel (11) verwenden wir nun zur **Schrittweitensteuerung**: Wir geben uns eine Toleranz $\tau > 0$ vor und bestimmen die Schrittweite in jedem Schritt so, dass gilt

$$\text{lokaler Fehler } \approx \tau. \tag{13}$$

Approximieren wir $\varepsilon(x_n, h)$ durch $\varepsilon(x_n, h) \approx \gamma h^2$, so kann man γ durch einen Probeschritt der Länge H schätzen:

$$\gamma \approx \frac{1}{H^2}\,\varepsilon(x_n, H).$$

Die optimale Wahl der Schrittweite ergäbe sich nach (13),

$$\tau = |\varepsilon(x_n, h)| \approx |\gamma| \cdot h^2 \approx \frac{h^2}{H^2}\,|\varepsilon(x_n, H)|,$$

also

$$h = H\sqrt{\frac{\tau}{|\varepsilon(x_n, H)|}}.$$

Der folgende MATLAB-Programmteil verwendet eine ähnliche Schrittweitenkontrolle bei gegebenen Startwerten x und y und gegebener Probeschrittlänge h:

```
v=1.e-5*ones(1,n);
z = f(x,y);
while h>0
  y1 = y + h*z;
  y2 = y + h/2*z;
  y2 = y2 + h/2 * f(x+h/2,y2);
  d=max(v,max(abs(y),abs(y1)));
  phi = 2 * norm((y2 - y1)./d);
  hneu = h * min(max(0.9*sqrt(tol/phi),0.2),10);
  if phi > tol
    h = hneu;
  else
    x = x + h;
    y = 2*y2 - y1;                (*)
    z = f(x,y);
    h = min(b-a,hneu);
    end
  end
```

Bemerkung 4 Es wurde der absolute Fehler durch den "relativen" Fehler ersetzt. Ferner wurde dabei der Betrag des Funktionswerts nach unten komponentenweise durch 10^{-5} begrenzt. Zusätzlich wurde die "optimale" Schrittweite *hneu* mit dem Faktor 0.9 verkleinert. Dies verringert die Wahrscheinlichkeit, daß der nächste Schritt verworfen wird. Schließlich wurde durch den minimalen Faktor 0.2 und den maximalen Faktor 10 dafür gesorgt, dass die Schrittweiten von Schritt zu Schritt sich nicht zu stark ändern. Die gewählten Konstanten 0.9, 0.2 und 10 lassen sich durch keine Theorie begründen, sondern haben sich in Codes bewährt. □

Bemerkung 5 Nach unserer Herleitung müsste in der Zeile $(*)$ $y = y1$ stehen. Da man aber ohne Mehrkosten die bessere Näherung $y = 2 * y2 - y1$ (Formel der Ordnung 2) zur Verfügung hat, verwendet man diese. Unsere Fehlerschätzung ist damit in der Regel pessimistisch. □

Aufgabe 2 Wieviele Funktionsauswertungen benötigt man, um die Anfangswertaufgabe

$$y' = y^2, \;\; y(0.8) = \frac{5}{6}, \;\; 0.8 \leq x \leq 1.8$$

mit $\tau = 0.001$ zu lösen? Wie groß ist der maximale Fehler, die maximale benutzte Schrittweite und die minimale benutzte Schrittweite?
Welche Schrittweite muss man wählen, um mit konstanter Schrittweite denselben maximalen Fehler zu erreichen? □

Das behandelte Polygonzugverfahren ist die einfachste Methode der großen Klasse der **Einschrittverfahren**, bei denen die Näherung y_{n+1} an dem neuen Punkt $x_{n+1} := x_n + h_n$ allein aus der Näherung y_n an der Stelle x_n und der Schrittweite h_n berechnet wird. Einschrittverfahren haben also die folgende Gestalt

$$y_{n+1} = y_n + h_n \, \Phi(x_n, y_n, h_n) \tag{14}$$

mit einer **Verfahrensfunktion** Φ.

Um die Güte von Einschrittverfahren zu beurteilen, führen wir die folgenden Begriffe ein:

Definition 6 *Es sei $z(x)$ die Lösung der Anfangswertaufgabe*

$$z' = f(x, z(x)), \;\; z(x_n) = y_n.$$

Dann heißt

$$\varepsilon(h) := z(x_n + h) - y_n - h \, \Phi(x_n, y_n, h)$$

der **lokale Fehler** *des durch (14) definierten Verfahrens.*

Das Verfahren (14) heißt **konsistent**, *falls $\varepsilon(h) = o(h)$ gilt, es heißt* **von der Ordnung** *p, wenn $\varepsilon(h) = O(h^{p+1})$ gilt.*

Dabei sind die **Landau Symbole** $o(h)$ und $O(h^q)$ folgendermaßen definiert:

$$\varepsilon(h) = o(h) \quad \Longleftrightarrow \quad \lim_{h \to 0+0} \frac{\varepsilon(h)}{h} = 0,$$

$$\varepsilon(h) = O(h^q) \quad \Longleftrightarrow \quad \text{es existiert } C > 0 \text{ mit } \frac{\varepsilon(h)}{h^q} \leq C.$$

Wie im Falle des Polygonzugverfahrens gilt:

Satz 7 *Es seien sie Näherungen y_n von $y(x_n)$ mit dem Einschrittverfahren*

$$y_{n+1} = y_n + h_n \Phi(x_n, y_n, h_n), \;\; n = 0, 1, \ldots, N,$$

berechnet.

Erfüllt die Verfahrensfunktion Φ eine Lipschitz Bedingung bzgl. y in $[a, b] \times \mathbb{R}$ (es genügt eine Umgebung der Lösung)

$$|\Phi(x, y, h) - \Phi(x, z, h)| \leq \Lambda |y - z| \tag{15}$$

und ist das Einschrittverfahren konsistent von der Ordnung p

$$|y(x+h) - y(x) - h\Phi(x, y(x), h)| \leq C \cdot h^{p+1}, \tag{16}$$

so gilt für den globalen Fehler

$$|\delta_n| = |y_n - y(x_n)| \leq C(b-a)e^{\Lambda(b-a)}\, h^p, \tag{17}$$

wobei

$$h := \max_{j=0,\ldots,N-1} h_j$$

gesetzt ist.

Bemerkung 8 Die Lipschitz Bedingung für die Verfahrensfunktion Φ erhält man in vielen Fällen aus der Lipschitz Bedingung für die rechte Seite f. □

Beispiel 9 (Polygonzugverfahren) *(Euler 1768)*
Das einfachste Einschrittverfahren ist das Eulersche Polygonzugverfahren

$$\Phi(x, y, h) = f(x, y).$$

□

Beispiel 10 Ist f differenzierbar, so erhält man wegen

$$y''(x) = \frac{d}{dx} f(x, y(x)) = f_x(x, y(x)) + f_y(x, y(x))y'(x) = f_x(x, y(x)) + f_y(x, y(x))f(x, y(x))$$

und

$$\begin{aligned} y(x_n + h) &= y(x_n) + hy'(x_n) + \frac{1}{2}h^2 y''(x_n) + O(h^3) \\ &= y_n + hf(x_n, y_n) + \frac{1}{2}h^2(f_x(x_n, y_n) + f_y(x_n, y_n)f(x_n, y_n)) + O(h^3) \end{aligned}$$

mit

$$\Phi(x, y, h) = f(x, y) + 0.5h(f_x(x, y) + f_y(x, y)f(x, y))$$

ein Einschrittverfahren der Ordnung 2. Es ist klar, wie man Verfahren höherer Ordnung finden kann, wenn f höhere Ableitungen besitzt.

Beispiel 11 (Verbessertes Polygonzugverfahren) *(Coriolis 1837, Runge 1895)*
Wir haben dieses Verfahren bereits durch Extrapolation aus dem Polygonzugverfahren mit den Schrittweiten h und $\frac{h}{2}$ hergeleitet (vgl. (18)):

$$y_{n+1} = y_n + h_n\, f\Big(x_n + \frac{h_n}{2}, y_n + \frac{h_n}{2}\, f(x_n, y_n)\Big)$$

und schon gesehen, dass das verbesserte Polygonzugverfahren die Ordnung 2 besitzt.

Geometrisch kann man dieses Verfahren so interpretieren: Es wird zunächst eine Schätzung $y_{n+\frac{1}{2}} = y_n + \frac{h_n}{2}\, f(x_n, y_n)$ für $y\Big(x_n + \frac{h_n}{2}\Big)$ ermittelt, und die hiermit berechnete Näherung $f\Big(x_n + \frac{h_n}{2}, y_{n+\frac{1}{2}}\Big) \approx y'\Big(x_n + \frac{h_n}{2}\Big)$ für die Steigung von y im ganzen Intervall $[x_n, x_n + h_n]$ verwendet. □

Beispiel 12 (Verfahren von Heun) *(Heun 1900)*
Man verwendet den Mittelwert zweier Steigungen

$$k_1 := f(x_n, y_n), \quad k_2 := f(x_n + h_n, y_n + h_n k_1)$$

und setzt hiermit

$$y_{n+1} = y_n + h_n \frac{k_1 + k_2}{2}.$$

Dieses Verfahren entspricht der Quadratur des Integrals in

$$y(x_{n+1}) - y(x_n) = \int_{x_n}^{x_{n+1}} f(t, y(t))\, dt$$

mit der Trapezregel, wenn man den unbekannten Punkt $(x_{n+1}, y(x_{n+1}))$ ersetzt durch $(x_{n+1}, y_n + h_n f(x_n, y_n))$.

Mit dem Taylorschen Satz kann man zeigen, dass für den lokalen Fehler

$$\varepsilon(h) = z(x_n + h) - y_n - \frac{h}{2}(f(x_n, y_n) + f(x_n + h, y_n + h\, f(x_n, y_n))) = O(h^3)$$

gilt, dass das Verfahren von Heun also wie das verbesserte Polygonzugverfahren die Ordnung 2 besitzt. □

Beispiel 13 (explizite Runge–Kutta Verfahren) *(Kutta 1901)*
Dies sind Verallgemeinerungen der Verfahren in Beispiel 9, Beispiel 11 und Beispiel 12 des folgenden Typs:

$$\begin{aligned} k_1 &:= f(x_n, y_n) \\ k_j &:= f(x_n + \alpha_j\, h_n, y_n + h_n \sum_{\ell=1}^{j-1} \beta_{j\ell}\, k_\ell), \quad j = 2, \ldots, s \\ y_{n+1} &:= y_n + h_n \sum_{j=1}^{s} \gamma_j\, k_j. \end{aligned} \tag{18}$$

Die Koeffizienten $\alpha_j, \beta_{j\ell}, \gamma_j$ werden dabei so gewählt, dass das Verfahren möglichst hohe Ordnung hat. s heißt die **Stufe** des Runge–Kutta Verfahrens. □

Wir betrachten zunächst zweistufige Runge–Kutta Verfahren

$$\begin{aligned} y_{n+1} &= y_n + h(\gamma_1 k_1 + \gamma_2 k_2) \\ &= y_n + h(\gamma_1 f(x_n, y_n) + \gamma_2 f(x_n + \alpha_2 h, y_n + h\beta_{21} f(x_n, y_n))). \end{aligned} \tag{19}$$

Wir bestimmen die Parameter γ_1, γ_2, α_2 und β_{21} so, dass das Verfahren möglichst große Ordnung hat.

Ist $f \in C^2$ (und damit $y \in C^3$), so gilt nach dem Taylorschen Satz

$$\begin{aligned} y(x+h) &= y(x) + hy'(x) + \frac{1}{2}h^2 y''(x) + \frac{1}{6}h^3 y'''(x) + o(h^3) \\ &= y(x) + hf(x, y(x)) + \frac{1}{2}h^2 (f_x(x, y(x)) + f_y(x, y(x)) f(x, y(x))) \\ &\quad + \frac{1}{6}h^3 [f_{xx} + 2f f_{xy} + f^2 f_{yy} + f_x f_y + f f_y^2](x, y(x)) + o(h^3) \end{aligned}$$

und

$$\begin{aligned} & y(x) + h(\gamma_1 f(x, y(x)) + \gamma_2 f(x + \alpha_2 h, y(x) + h\beta_{21} f(x, y(x)))) \\ = \;& y(x) + h\gamma_1 f(x, y(x)) + h\gamma_2 [f + \alpha_2 h f_x + \beta_{21} h f_y f + \frac{1}{2}(\alpha_2 h)^2 f_{xx} \\ & + \alpha_2 \beta_{21} h^2 f_{xy} f + \frac{1}{2}(\beta_{21} h)^2 f^2 f_{yy}](x, y(x)) + o(h^3). \end{aligned}$$

Subtraktion und Ordnen nach Potenzen von h liefert

$$\begin{aligned} & y(x+h) - y(x) - h(\gamma_1 f(x, y(x)) + \gamma_2 f(x + \alpha_2 h, y(x) + h\beta_{21} f(x, y(x)))) \\ = \;& h(1 - \gamma_1 - \gamma_2) f(x, y(x)) \\ & + \frac{1}{2} h^2 [(1 - 2\gamma_2 \alpha_2) f_x + (1 - 2\gamma_2 \beta_{21}) f f_y](x, y(x)) \\ & + \frac{1}{6} h^3 [(1 - 3\gamma_2 \alpha_2^2) f_{xx} + 2(1 - 3\gamma_2 \alpha_2 \beta_{21}) f_{xy} f \\ & \quad + (1 - 3\gamma_2 \beta_{21}^2) f_{yy} f^2 + f_x f_y + f f_y^2](x, y(x)) + o(h^3). \end{aligned}$$

Da bei keiner Wahl der Parameter der Koeffizient bei h^3 für alle Funktionen f verschwindet, können wir keine höhere Konvergenzordnung als 2 erreichen.

Für Verfahren der Ordnung 2 muss gelten

$$\gamma_1 + \gamma_2 = 1, \; 2\gamma_2 \alpha_2 = 1, \; 2\gamma_2 \beta_{21} = 1. \tag{20}$$

Dieses System von 3 Gleichungen in 4 Unbekannten besitzt unendlich viele Lösungen. Wählt man γ_2 als freien Parameter, so erhält man die Lösungsschar

$$\gamma_1 = 1 - \gamma_2, \; \alpha_2 = \beta_{21} = \frac{1}{2\gamma_2}, \; \gamma_2 \neq 0. \tag{21}$$

Die bereits betrachteten Verfahren der Ordnung 2 sind hierin enthalten. Für $\gamma_1 = 1$ erhält man das verbesserte Polygonzugverfahren, für $\gamma_2 = 0.5$ das Verfahren von Heun.

Bevor wir Verfahren größerer Ordnung als 2 angeben, schicken wir einige Bemerkungen voraus über die mit einem s-stufigen Verfahren erreichbare Konsistenzordnung. Diese Frage ist keinesfalls leicht zu beantworten, da die beim Taylorabgleich entstehenden Bedingungsgleichungen nichtlinear in den Parametern sind. Eine sorgfältige Untersuchung mit Hilfe von Ordnungsbäumen findet man in **Hairer, Nørsett und Wanner** [2].

Die Zahl der zu erfüllenden Gleichungen steigt mit wachsender Ordnung p sehr stark an, wie die folgende Tabelle zeigt:

Ordnung p	1	2	3	4	5	6	7	8	9	10
Zahl der Gleichungen	1	2	4	8	17	37	85	200	486	1205.

Dabei wurden die Gleichungen

$$\sum_{j=1}^{k-1} \beta_{kj} = \alpha_k, \; k = 2, \ldots, m,$$

die wir stets als erfüllt annehmen, nicht mitgezählt.

Gibt man die Ordnung p vor und bestimmt dazu die Stufenzahl s des Runge-Kutta-Verfahrens minimal, so gilt der folgende Zusammenhang

p	1	2	3	4	5	6	7	8	9	10
s	1	2	3	4	6	7	9	10	11	12.

Man sieht, dass sich mit wachsender Ordnung das Verhältnis von erreichbarer Ordnung p zur Zahl der dazu nötigen Stufen (also zur Zahl der Funktionsauswertungen in jedem Schritt) verschlechtert.

Entwickelt man (18) bis zu Termen mit h^2, so sieht man unmittelbar:

Satz 14 *Das Einschrittverfahren (18) ist genau dann konsistent, wenn gilt*

$$\sum_{j=1}^{s} \gamma_j = 1.$$

Wir geben nun die wichtigsten Runge–Kutta Formeln an. Dazu verwendet man meistens die folgende Tableau Darstellung:

$$\begin{array}{c|cccc} 0 & & & & \\ \alpha_2 & \beta_{21} & & & \\ \alpha_3 & \beta_{31} & \beta_{32} & & \\ \vdots & & & & \\ \alpha_s & \beta_{s1} & \beta_{s2} & \cdots & \beta_{s,s-1} \\ \hline & \gamma_1 & \gamma_2 & \cdots & \gamma_{s-1} \quad \gamma_s. \end{array}$$

Die uns bekannten Verfahren der Ordnung 2 kann man damit so schreiben

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

Verfahren von Heun

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

verb. Polygonzugverfahren

Am bekanntesten ist das **klassische Runge-Kutta-Verfahren** (1895) der Ordnung 4.

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

Ausführlich geschrieben lautet dieses

$$\begin{aligned} k_1 &= f(x_n, y_n) \\ k_2 &= f\Big(x_n + \frac{h_n}{2}, y_n + \frac{h_n}{2}\, k_1\Big) \\ k_3 &= f\Big(x_n + \frac{h_n}{2}, y_n + \frac{h_n}{2}\, k_2\Big) \\ k_4 &= f(x_n + h_n, y_n + h_n\, k_3) \\ y_{n+1} &= y_n + h_n\, \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}. \end{aligned}$$

Ein weiteres Verfahren der Ordnung 4 ist die **3/8–Regel** (Kutta 1901)

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{3} & \frac{1}{3} & & & \\ \frac{2}{3} & -\frac{1}{3} & 1 & & \\ 1 & 1 & -1 & 1 & \\ \hline & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{array} .$$

Die 3/8-Regel verallgemeinert die Newtonsche 3/8-tel Regel (oder Keplersche Fassregel oder pulcherima) der numerischen Integration.

Die klassische Runge–Kutta Regel ist das bekannteste, die 3/8-Regel häufig das genaueste der expliziten Runge–Kutta Verfahren der Ordnung 4.

Aufgabe 3 Bestimmen Sie für die Anfangswertaufgabe

$$y' = y^2,\ y(0.8) = \frac{5}{6},\ x \in [0.8, 1.8],$$

mit dem verbesserten Polygonzugverfahren, dem Verfahren von Heun, dem klassischen Runge–Kutta Verfahren und der 3/8–Regel Näherungen für die Schrittweiten $h = \frac{1}{20}$, $h = \frac{1}{40}$, $h = \frac{1}{80}$, $h = \frac{1}{160}$, $\frac{1}{320}$ und $\frac{1}{640}$. Bestätigen Sie, dass die ersten beiden Verfahren die Ordnung 2 besitzen und die letzten beiden die Ordnung 4. □

Eine Schrittweitensteuerung kann man für die Runge–Kutta Verfahren prinzipiell wie für das Polygonzugverfahren durchführen. Um den Fehler zu schätzen, kann man zwei Schritte mit der halben Schrittweite ausführen. Im Falle der klassischen Runge-Kutta-Verfahren hat man dabei die Funktion f an 7 zusätzlichen Punkten auszuwerten, so dass man in jedem Schritt insgesamt 11 Funktionsauswertungen benötigt.

Mit wesentlich weniger Aufwand kommt man bei den **eingebetteten Runge-Kutta-Formeln** aus:

Die Idee ist — ähnlich wie bei den Kronrod-Formeln zur Quadratur — von einer Runge-Kutta-Formel der Stufe s mit den Zuwächsen $k_1, \ldots, k_s$ und der Ordnung p auszugehen und hierzu bei erhöhter Stufenzahl σ weitere $k_{s+1}, \ldots, k_\sigma$ zu bestimmen, so dass die Formel

$$\tilde{y}_{n+1} = y_n + h_n\Big(\sum_{j=1}^{s} \tilde{\gamma}_j\, k_j + \sum_{j=s+1}^{\sigma} \tilde{\gamma}_j\, k_j\Big)$$

eine höhere Ordnung q als die Ausgangsformel hat.

Dann gilt für die lokalen Fehler $\varepsilon = C\, h^{p+1} + O(h^{p+2})$ und $\tilde{\varepsilon} = O(h^{q+1}) = O(h^{p+2})$, d.h. $\tilde{y}_{n+1} - y_{n+1} = C\, h^{p+1} + O(h^{p+2})$, und hiermit kann man bei vorgegebener Toleranz die optimale Schrittweite wie vorher schätzen.

In der Literatur wurde eine Reihe von Formelpaaren angegeben. Das derzeit bevorzugte Paar der Ordnungen 5 und 4 geht auf **Dormand und Prince** zurück:

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	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
$p = 5$	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
$q = 4$	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$.

Aufgabe 4 Lösen Sie die Anfangswertaufgabe

$$y' = y^2, \;\; y(0.8) = \frac{5}{6}, \;\; x \in [0.8, 1.8]$$

mit dem Verfahren von Dormand und Prince, so dass der Fehler höchstens $1e-6$ ist. Vergleichen Sie dessen Aufwand mit dem des klassischen Runge–Kutta Verfahrens mit konstanter Schrittweite. □

FORTRAN77 Codes der Methoden von Dormand und Prince sind in dem Buch von **Hairer, Nørsett, Wanner** [2] abgedruckt. Man kann diese (oder auch C Versionen) erhalten unter

http://www.unige.ch/math/folks/hairer/software.html

In MATLAB 5.2 sind als eingebettete Runge–Kutta Methoden für nicht-steife Anfangswertaufgaben das Verfahren von Dormand und Prince der Ordnung 5 als ODE45 und für den Fall, dass keine sehr hohe Genauigkeit benötigt wird, das Verfahren von Bogacki und Shampine (vgl. **Shampine** [4]) der Ordnung 2 als ODE23 implementiert.

Weitere Software findet man in der Spiegelung der NETLIB in der elib des ZIB

http://elib.zib.de/netlib

in den Unterverzeichnissen 'ode' und 'odepack'.

Im ZIB wird das ODELab vorbereitet. Es enthält 7 Löser für Anfangswertaufgaben und 16 Beispiele, an denen die Verfahren getestet werden können. Das ODELab ist unter

http://galois.zib.de:8001/public/cgi-bin/odelab

bereits zugänglich. Es fehlt bisher allerdings jede Dokumentation. In Zukunft soll man neben den 16 Standardbeispielen auch eigene Beispiele verwenden können.

3 Mehrschrittverfahren

Ein weiterer, häufig benutzter Verfahrenstyp zur numerischen Lösung der Anfangswertaufgabe

$$y' = f(x, y), \;\; y(a) = y_0$$

sind die linearen **Mehrschrittverfahren**, bei denen man zur Berechnung der Näherung y_{n+k} die bereits ermittelten Näherungen $y_{n+k-1}, y_{n+k-2}, \ldots, y_n$ verwendet. Dazu macht man den Ansatz

$$\sum_{\nu=0}^{k} a_\nu \, y_{n+\nu} = h \sum_{\nu=0}^{k} b_\nu \, f_{n+\nu} \tag{22}$$

mit $f_{n+\nu} := f(x_{n+\nu}, y_{n+\nu})$, wobei $a_k \neq 0$ vorausgesetzt wird.

Ist $b_k \neq 0$, so kommt y_{n+k} auf beiden Seiten von (22) vor. Es muss dann in jedem Schritt ein (i.a. nichtlineares) Gleichungssystem gelöst werden, um y_{n+k} zu bestimmen. In diesem

Fall heißt (22) **implizites k-Schritt Verfahren**. Ist $b_k = 0$, so kann man (22) sofort nach y_{n+k} auflösen. In diesem Fall heißt (22) **explizites k-Schritt Verfahren**.

Offenbar ist der erste Wert, den man mit (22) berechnen kann, y_k. Neben dem gegebenen Wert y_0 müssen also zunächst Näherungen $y_1, \ldots, y_{k-1}$ für $y(x_1), \ldots, y(x_{k-1})$ zur Verfügung gestellt werden. Diese können z.B. mit einem Runge-Kutta-Verfahren berechnet werden.

Wegen $a_k \neq 0$ können wir o.B.d.A. $a_k = 1$ annehmen. Wir bestimmen die übrigen a_ν, b_ν nun so, dass (22) zu einem brauchbaren Verfahren wird.

Den lokalen Fehler von (22) erhält man wieder, indem man die exakte Lösung $y(x)$ von (1) in (22) einsetzt:

$$\varepsilon_n := \sum_{\nu=0}^{k} a_\nu \, y(x_n + \nu h) - h \sum_{\nu=0}^{k} b_\nu \, y'(x_n + \nu h). \tag{23}$$

Ist y $(m+1)$-mal differenzierbar, so liefert der Taylorsche Satz

$$\begin{aligned} \varepsilon_n \;=\; & \sum_{\nu=0}^{k} a_\nu \Big(\sum_{\mu=0}^{m} \frac{y^{(\mu)}(x_n)}{\mu!} \, \nu^\mu h^\mu + \frac{1}{(m+1)!} \, y^{(m+1)}(x_n + \theta_\nu \nu h) \, \nu^{m+1} h^{m+1} \Big) \\ & - \sum_{\nu=0}^{k} b_\nu \Big(\sum_{\mu=0}^{m-1} \frac{y^{(\mu+1)}(x_n)}{\mu!} \, \nu^\mu h^{\mu+1} + \frac{1}{m!} \, y^{(m+1)}(x_n + \hat{\theta}_\nu \nu h) \, \nu^m h^{m+1} \Big). \end{aligned}$$

Damit das Verfahren konsistent ist, müssen sich die Glieder mit dem Faktor h^0 und mit dem Faktor h^1 jeweils gegenseitig aufheben, d.h. es muss gelten

$$\sum_{\nu=0}^{k} a_\nu = 0, \quad \sum_{\nu=0}^{k} (\nu a_\nu - b_\nu) = 0. \tag{24}$$

Das Nächstliegende ist nun, die a_ν, b_ν so zu bestimmen, dass in ε_n möglichst hohe Potenzen von h abgeglichen werden.

Dies führt zu einem linearen (wegen $a_k = 1$ inhomogenen) Gleichungssystem.

Für $k = 2$ erhält man

$$\begin{array}{rcrcrcrcll} a_0 & + & a_1 & & & & & = & -1 & \text{(siehe (24))} \\ & & a_1 & - & b_0 & - & b_1 & - & b_2 = & -2 \quad \text{(siehe (24))} \\ & & a_1 & & & - & 2b_1 & - & 4b_2 = & -4 \\ & & a_1 & & & - & 3b_1 & - & 12b_2 = & -8 \\ & & a_1 & & & - & 4b_1 & - & 32b_2 = & -16 \end{array} \tag{25}$$

mit der Lösung $a_0 = -1$, $a_1 = 0$, $b_0 = \dfrac{1}{3}$, $b_1 = \dfrac{4}{3}$, $b_2 = \dfrac{1}{3}$.

Man erhält also das implizite Verfahren der Ordnung 4:

$$y_{n+2} = y_n + \frac{h}{3} (f_n + 4 f_{n+1} + f_{n+2}).$$

Verlangt man, um die Auflösung einer nichtlinearen Gleichung (bzw. eines nichtlinearen Gleichungssystems) in y_{n+2} in jedem Schritt zu vermeiden, $b_2 = 0$, d.h. ein explizites Verfahren, so kann man nur die ersten vier Gleichungen von (25) erfüllen. Lösung hiervon ist $a_0 = -5$, $a_1 = 4$, $b_0 = 2$, $b_1 = 4$, und man erhält das explizite Verfahren der Ordnung 3:

$$y_{n+2} = -4 y_{n+1} + 5 y_n + 2h (f_n + 2 f_{n+1}). \tag{26}$$

Tabelle 1: AWA mit Anfangsfehler 10^{-15} ($y \equiv 1$)

n	y_n	n	y_n
0	1.00000000000000000	16	0.99997176556842504
1	1.00000000000000111	17	1.00014117215787590
2	0.99999999999999556	18	0.99929413921062160
3	1.00000000000002331	19	1.00352930394689310
4	0.99999999999988454	20	0.98235348026553560
5	1.00000000000057843	21	1.08823259867232314
6	0.99999999999710898	22	0.55883700663838543
7	1.00000000001445621	$\vdots$	.....
8	0.99999999992772004		
9	1.00000000036140091	34	$-1.077058079E+0008$
10	0.99999999819299656	35	$5.385290456E+0008$

Wendet man (26) auf die Anfangswertaufgabe $y' = 0$, $y(0) = 1$ mit dem (z.B. durch Rundungsfehler verfälschten) Anfangsfeld $y_0 = 1$, $y_1 = 1+10^{-15}$ an, so erhält man die Werte in Tabelle 1. Kleinste Anfangsfehler schaukeln sich also auf und machen das Verfahren trotz der Ordnung 3 völlig unbrauchbar.

Die Fehlerordnung allein ist damit (anders als bei Einschrittverfahren) kein geeignetes Mittel zur Bewertung eines Mehrschrittverfahrens.

Für den Fall $f(x, y) \equiv 0$ lautet (26)

$$y_{n+2} + 4y_{n+1} - 5y_n = 0. \tag{27}$$

Dies ist eine lineare homogene Differenzengleichung mit konstanten Koeffizienten. Der Ansatz $y_n = \lambda^n$ für eine Lösung von (27) führt auf die Bedingung

$$\lambda^{n+2} + 4\lambda^{n+1} - 5\lambda^n = 0$$

d.h. $\lambda^2 + 4\lambda - 5 = 0$ mit den Lösungen $\lambda_1 = 1$, $\lambda_2 = -5$.

Da (27) linear und homogen ist, ist auch

$$y_n = A\lambda_1^n + B\lambda_2^n = A + B(-5)^n, \quad A, B \in \mathbb{R},$$

eine Lösung (sogar die allgemeine Lösung). Die Konstanten A und B kann man aus dem Anfangsfeld y_0 und y_1 bestimmen. Man erhält

$$y_n = \frac{1}{6}(5y_0 + y_1) + (-5)^n \frac{1}{6}(y_0 - y_1).$$

Der zweite Term hiervon führt dazu, dass sich die Fehler (bei alternierendem Vorzeichen) aufschaukeln.

Im allgemeinen Fall (22) hätte man statt (27) für $f(x, y) \equiv 0$ die Differenzengleichung $\sum_{\nu=0}^{k} a_\nu \, y_{n+\nu} = 0$ mit der charakteristischen Gleichung

$$\rho(\lambda) := \sum_{\nu=0}^{k} a_\nu \, \lambda^\nu = 0.$$

Sind $\lambda_1, \ldots, \lambda_r$ die verschiedenen Nullstellen von ρ mit den Vielfachheiten $m_1, \ldots, m_r$, so sind alle Lösungen von

$$\sum_{\nu=0}^{n} a_\nu \, y_{n+\nu} = 0$$

Linearkombinationen von $\lambda_j^n, n\lambda_j^n, \ldots, n^{m_j-1}\lambda_j^n$, $j = 1, \ldots, r$.

Fehler im Anfangsfeld $y_0, \ldots, y_{k-1}$ werden daher nicht verstärkt, wenn $|\lambda_j| \leq 1$ für alle Nullstellen λ_j von ρ gilt und die Nullstellen mit $|\lambda_j| = 1$ einfach sind.

Definition 15 *Das k-Schritt Verfahren (22) heißt* **stabil**, *wenn für alle Nullstellen λ_j des charakteristischen Polynoms*

$$\rho(\lambda) := \sum_{\nu=0}^{k} a_\nu \lambda^\nu$$

$|\lambda_1| \leq 1$ *gilt und wenn die Nullstellen mit $|\lambda_j| = 1$ einfach sind.*

Wegen der Konsistenzbedingung ist stets $\lambda = 1$ eine Nullstelle von ρ. Gilt $|\lambda_j| < 1$ für alle anderen Nullstellen λ_j von ρ, so heißt das Verfahren **stark stabil**.

Die obigen Überlegungen zeigen, dass die Stabilität neben der Konsistenz die Mindestanforderung an ein k-Schritt Verfahren ist. Umgekehrt kann man zeigen, dass konsistente, stabile Verfahren konvergieren.

Fordert man in (25) neben $b_2 = 0$ (**Explizitheit**), dass $\rho(\lambda)$ die Nullstellen $\lambda_1 = 1$ (**Konsistenz**) und $\lambda_2 = 0$ (um die **Stabilität** zu erzwingen) besitzt, so kann man nur die ersten drei Gleichungen von (25) erfüllen und erhält das explizite Verfahren der Ordnung 2:

$$y_{n+2} = y_{n+1} + \frac{h}{2}(-f_n + 3f_{n+1}).$$

Wir geben nun einen Weg an, wie man stark stabile Mehrschrittverfahren konstruieren kann.

Für die Lösung y der Anfangswertaufgabe (1) gilt

$$y(x_{n+1}) - y(x_n) = \int_{x_n}^{x_{n+1}} y'(t)\,dt = \int_{x_n}^{x_{n+1}} f(t, y(t))\,dt. \tag{28}$$

Wir ersetzen daher bei gegebenen Näherungen $y_j \approx y(x_j)$, $j = n, n-1, \ldots, n-k+1$, und damit bekannten Näherungen

$$f_j := f(x_j, y_j) \approx f(x_j, y(x_j)) = y'(x_j),\ j = n, n-1, \ldots, n-k+1,$$

die Funktion y' im Integranden durch ihr Interpolationspolynom

$$p \in \Pi_{k-1}\ :\ p(x_j) = f_j,\ j = n, n-1, \ldots, n-k+1,$$

und berechnen die neue Näherung gemäß

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} p(t)\,dt.$$

Mit der Lagrangeschen Integrationsformel kann man nachrechnen (vgl. [7]), daß y_{n+1} sich berechnen läßt als

$$y_{n+1} = y_n + h \cdot \sum_{j=0}^{k-1} \alpha_j f_{n-j}, \tag{29}$$

wobei die Koeffizienten α_j unabhängig von den y_j und von den speziellen Knoten x_j und der Schrittweite h sind, und daher in Dateien bereitgestellt werden können.

Das charakteristische Polynom des Verfahrens (29) ist

$$\rho(\lambda) = \lambda^k - \lambda^{k-1}$$

mit der einfachen Nullstelle $\lambda = 1$ und der $(k-1)$-fachen Nullstelle 0. Die Mehrstellenformel ist also stark stabil.

So konstruierte Mehrstellenformeln heißen **Adams–Bashforth Verfahren**. Sie sind explizit und aus der Fehlerdarstellung des Interpolationspolynoms erhält man, dass ihre Ordnung k ist. Die ersten Adams–Bashforth Formeln sind:

$$\begin{aligned}
k = 1 \quad & y_{n+1} = y_n + h f_n \\
k = 2 \quad & y_{n+1} = y_n + 0.5h(3f_n - f_{n-1}) \\
k = 3 \quad & y_{n+1} = y_n + h(23f_n - 16f_{n-1} + 5f_{n-2})/12 \\
k = 4 \quad & y_{n+1} = y_n + h(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})/24.
\end{aligned}$$

Nachteil der Adams–Bashforth Formeln ist, dass bei ihrer Konstruktion das Interpolationspolynom p im Intervall $[x_n, x_{n+1}]$ verwendet wird, während die Interpolationsknoten außerhalb dieses Intervalls liegen. Wir wissen bereits, dass der Fehler eines Interpolationspolynoms außerhalb des kleinsten Intervalls $[x_{n-k+1}, x_n]$, das alle Knoten enthält, sehr schnell anwächst. Es ist daher naheliegend, die Funktion y' in (28) durch das Interpolationspolynom

$$p \in \Pi_k \ : \ p(x_j) = f(x_j, y_j), j = n+1, n, n-1, \ldots, n-k+1$$

zu ersetzen.

Wie eben kann man das Verfahren schreiben als

$$y_{n+1} = y_n + h \sum_{j=0}^{k} \beta_j f_{n+1-j}$$

mit

$$\beta_j := \frac{1}{h} \int_{x_n}^{x_{n+1}} \prod_{\substack{i=0 \\ i \neq j}}^{k} (t - x_{n+1-i}) \Big/ \prod_{\substack{i=0 \\ i \neq j}}^{k} (x_{n+1-j} - x_{n+1-i})\, dt.$$

Diese Verfahren heißen **Adams–Moulton Verfahren**. Sie sind wie die Adams–Bashforth Verfahren stark stabil und haben die Ordnung $k+1$ (Beachten Sie, dass der Grad des Interpolationspolynoms hier k ist, beim Adams–Bashforth Verfahren aber nur $k-1$). Die ersten Adams-Moulton Formeln sind:

$$\begin{aligned}
k = 0 \quad & y_{n+1} = y_n + h f_{n+1} \\
k = 1 \quad & y_{n+1} = y_n + 0.5h(f_{n+1} + f_n) \\
k = 2 \quad & y_{n+1} = y_n + h(5f_{n+1} + 8f_n - f_{n-1})/12 \\
k = 3 \quad & y_{n+1} = y_n + h(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2})/24.
\end{aligned}$$

Die Adams–Moulton Verfahren haben wesentlich bessere Konvergenzeigenschaften als die Adams–Bashforth Verfahren gleicher Ordnung. Nachteilig ist, dass sie implizit sind (beachten Sie $f_{n+1} = f(x_{n+1}, y_{n+1})$), man also in jedem Schritt ein nichtlineares Gleichungssystem zu lösen hat.

Man kombiniert beide Verfahren zu einem **Prädiktor-Korrektor-Verfahren**: Sind bereits Näherungen $y_j = y(x_j)$, $j = 0, \ldots, n$, bekannt $(n \geq k)$, so bestimme man mit dem Adams–Bashforth Verfahren der Ordnung k eine vorläufige Näherung

$$\tilde{y}_0 := y_n + h \sum_{j=0}^{k} \alpha_j f_{n-j}$$

für $y(x_{n+1})$ und verbessere diese iterativ unter Benutzung der Adams–Moulton Formel der Ordnung $k+1$:

$$\tilde{y}_{i+1} = y_n + h\Big(\beta_0 f(x_{n+1}, \tilde{y}_i) + \sum_{j=1}^{k} \beta_j f_{n+1-j}\Big),\ i = 0, 1, \ldots.$$

Erfüllt f eine Lipschitz Bedingung und ist h genügend klein gewählt, so ist diese Iteration konvergent. In der Regel genügen ein oder zwei Verbesserungsschritte (sonst ist die Schrittweite h zu groß). Das so gefundene $\tilde{y}_1$ oder $\tilde{y}_2$ wird als y_{n+1} gewählt und es wird der nächste Prädiktor-Korrektor-Schritt ausgeführt.

Eine typische Implementierung eines Prädiktor-Korrektor Verfahrens hat die folgende Gestalt: P (Auswertung der Prädiktorformel) E (Evaluation von f) C (Auswertung der Korrektorformel) E (Evaluation von f) oder PECECE. Mit den Adams-Bashforth und Adams-Moulton Formeln für $k = 1$ (dem expliziten Euler Verfahren und der Mittelpunktregel) erhält man als PECECE Verfahren

$$\begin{aligned}
P \quad & y_{n+1}^{[1]} = y_n + h f_n \\
E \quad & f_{n+1}^{[1]} = f(x_{n+1}, y_{n+1}^{[1]}) \\
C \quad & y_{n+1}^{[2]} = y_n + 0.5h(f_{n+1}^{[1]} + f_n) \\
E \quad & f_{n+1}^{[2]} = f(x_{n+1}, y_{n+1}^{[2]}) \\
C \quad & y_{n+1} = y_n + 0.5h(f_{n+1}^{[2]} + f_n) \\
E \quad & f_{n+1} = f(x_{n+1}, y_{n+1}^{[2]}).
\end{aligned}$$

Vorteil der Mehrschrittverfahren ist, dass auch bei größeren Ordnungen nur in jedem Schritt eine Funktionsauswertung von f im expliziten Fall bzw. 2 oder 3 Auswertungen beim Prädiktor-Korrektor-Verfahren benötigt werden, während beim Einschrittverfahren die Zahl der Funktionsauswertungen bei Steigerung der Ordnung sehr rasch wächst. Mehrschrittverfahren werden daher vor allem verwendet, wenn die Auswertung von f sehr teuer ist.

Nachteil der Mehrschrittverfahren ist, dass die Schrittweitensteuerung komplizierter als beim Einschrittverfahren ist. Man muss

- entweder nicht äquidistante Knoten $x_n, x_{n-1}, \ldots, x_{n-k+1}$ verwenden und kann dann die α_j bzw. β_j nicht einer Tabelle entnehmen, sondern muss sie nach jeder Veränderung der Schrittweite während der nicht äquidistanten Phase neu berechnen
- oder bei geänderter Schrittweite $\tilde{h}$ Näherung $\tilde{y}_{n-j}$ für $y(x_n - j \cdot \tilde{h})$ aus einem Interpolationspolynom berechnen.

Der zweite Zugang wird in Hairer, Nørsett, Wanner [2] zusammen mit einer kombinierten Schrittweiten- und Ordnungskontrolle diskutiert.

In MATLAB 5.2 ist als Funktion ODE113 ein PECE Verfahren von Adams-Bashforth-Moulton zur Lösung von nicht-steifen Anfangswertaufgaben implementiert.

Die bisherigen Mehrschrittverfahren basierten auf der numerischen Lösung der Integralgleichung (28). Die folgende Klasse von Verfahren wird mit Hilfe der numerischen Differentiation konstruiert.

Es seien bereits Näherungen $y_{n-k+1}, \ldots, y_n$ der Lösung der Anfangswertaufgabe (1) an den Knoten $x_{n-k+1}, \ldots, x_n$ bekannt. Um $y_{n+1} \approx y(x_{n+1})$ zu erhaltenen, betrachten wir das Interpolationspolynom q zu den Daten (x_{n-k+1}, y_{n-k+1}), $\ldots$, (x_n, y_n), (x_{n+1}, y_{n+1}), und bestimmen y_{n+1} so, dass das Polynom q die Differentialgleichung an einem Gitterpunkt erfüllt:

$$q'(x_{n-1+\ell}) = f(x_{n-1+\ell}, y_{n-1+\ell}),\ \ell \in \{0, 1, \ldots\}. \tag{30}$$

Für $\ell = 1$ erhält man explizite Formeln, und zwar für $k = 1$ das explizite Euler Verfahren und für $k = 2$ die Mittelpunktregel. Die Formeln für $k \geq 3$ sind instabil und daher wertlos.

Für $\ell = 0$ erhält man implizite Formeln, die **BDF–Methoden** (backward differentiation formulas)

$$\begin{aligned}
k = 1 &: \ y_{n+1} - y_n = hf_{n+1} \\
k = 2 &: \ 3y_{n+1} - 4y_n + y_{n-1} = 2hf_{n+1} \\
k = 3 &: \ 11y_{n+1} - 18y_n + 9y_{n-1} - 2y_{n-2} = 6hf_{n+1} \\
k = 4 &: \ 25y_{n+1} - 48y_n + 36y_{n-1} - 16y_{n-2} + 3y_{n-3} = 12hf_{n+1} \\
k = 5 &: \ 137y_{n+1} - 300y_n + 300y_{n-1} - 200y_{n-2} + 75y_{n-3} - 12y_{n-4} = 60hf_{n+1} \\
k = 6 &: \ 147y_{n+1} - 360y_n + 450y_{n-1} - 400y_{n-2} + 225y_{n-3} - 72y_{n-4} + 10y_{n-5} = 60hf_{n+1}.
\end{aligned}$$

Durch Diskussion des Polynoms

$$\rho(\lambda) = \sum_{j=1}^{k} \frac{1}{j} \lambda^{k-j} (\lambda - 1)^j$$

sieht man für $k \leq 6$ leicht, dass diese BDF–Formeln stabil sind. Für $k \geq 7$ sind sie instabil (vgl. **Hairer–Nørsett–Wanner** [2], p. 380).

4 Steife Probleme

Es gibt Differentialgleichungen mit Lösungen, zu deren Approximation bei Anwendung expliziter Verfahren viel kleinere Schrittweiten benötigt werden, als man erwartet. Diese Probleme werden **steif** genannt.

Aufgabe 5 Die Anfangswertaufgabe

$$y' = -\lambda(y - e^{-x}) - e^{-x}, \quad y(0) = 1 \tag{31}$$

besitzt für alle $\lambda \in \mathbb{R}$ die eindeutige Lösung $y(x) = e^{-x}$.
Bestimmen Sie für $\lambda = 1$ und $\lambda = 10000$ Näherungslösungen mit dem Polygonzugverfahren, dem verbesserten Polygonzug Verfahren und dem klassischen Runge–Kutta Verfahren mit der konstanten Schrittweite $h = 0.001$. Wie beurteilen Sie den Erfolg der Verfahren? □

Abbildung 2 und Abbildung 3 zeigen die Lösungen der Anfangswertaufgaben

$$y' = -\lambda(y - e^{-x}) - e^{-x},\ y(x_0) = y_0$$

für verschiedene Werte von x_0 und y_0 für $\lambda = 1$ und $\lambda = 20$.

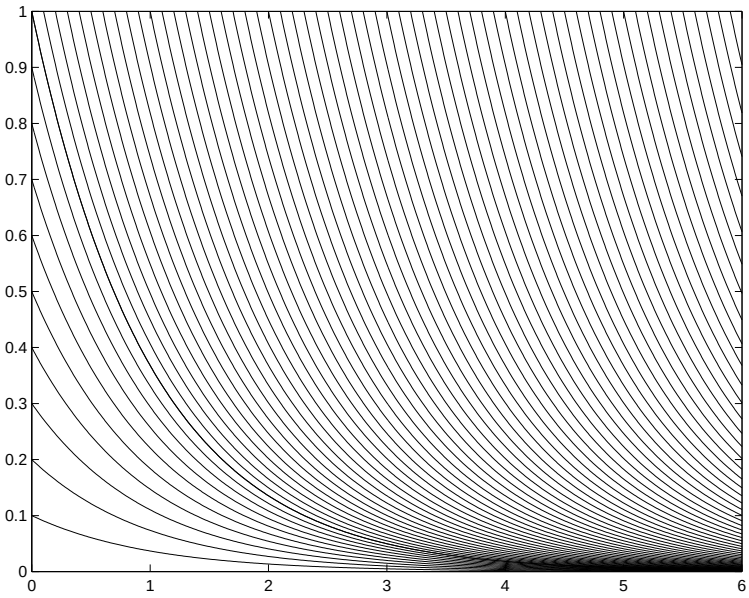

Abbildung 2: Lösungen für $\lambda = 1$

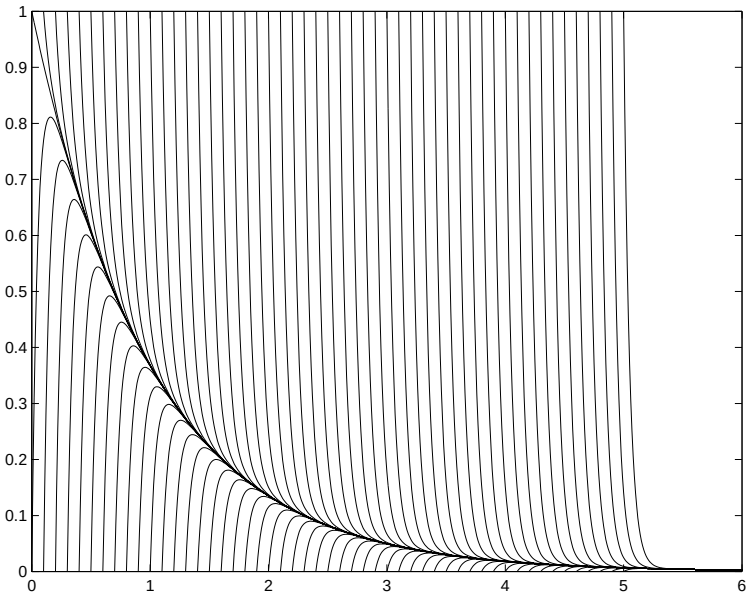

Abbildung 3: Lösungen für $\lambda = 20$

Wir möchten die Lösung $y(x) = e^{-x}$ von (31) für großes λ mit einem numerischen Verfahren verfolgen. Durch (Rundungs- oder Verfahrens-) Fehler werden wir ein wenig von der Lösung weggeführt. Ist $y_n \neq e^{-x_n}$, so konvergiert die Lösung

$$y(x; x_n, y_n) = (y_n - e^{-x_n})e^{-\lambda(x-x_n)} + e^{-x}$$

der Anfangswertaufgabe für großes λ sehr rasch gegen die quasi stationäre Lösung $\tilde{y}(x) = e^{-x}$. Die dem Betrage nach sehr große Steigung

$$y'(x_n; x_n, y_n) = -\lambda(y_n - e^{-x_n}) - e^{-x_n}$$

führt dazu, dass für das Euler Verfahren (und für die anderen Einschrittverfahren genauso) über das Ziel hinausgeschossen wird und die Fehler sich aufschaukeln.

Wendet man das Eulersche Polygonzugverfahren auf die Testgleichung

$$y' = \lambda y, \ \lambda < 0, \tag{32}$$

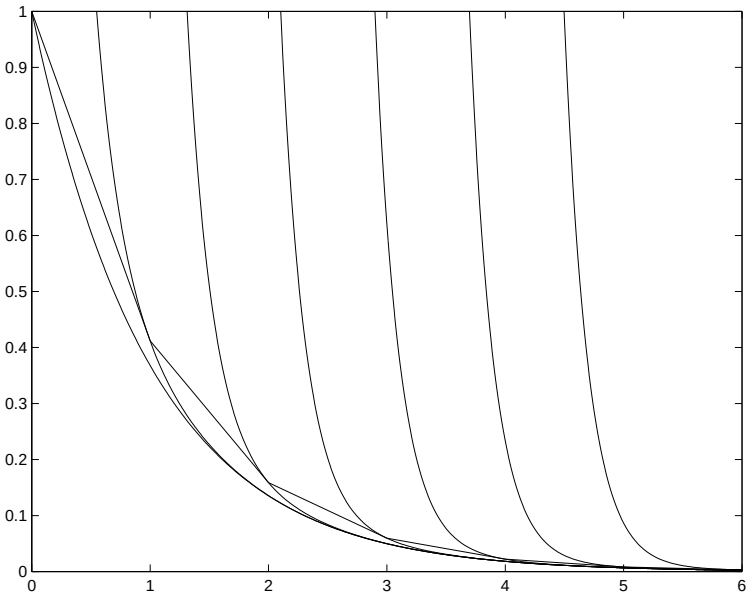

Abbildung 4: Implizites Euler Verfahren

an, so erhält man $y_{n+1} = y_n + h\lambda y_n$, und daher bei konstanter Schrittweite $h > 0$

$$y_n = (1 + h\lambda)^n y_0.$$

Für $|1+h\lambda| > 1$ explodiert die numerische Lösung y_n, und zwar ist dieses Aufschaukeln um so rascher, je kleiner λ ist, je schneller die Lösung der Anfangswertaufgabe also abklingt. Das Mindeste, was man von einem Verfahren erwarten muss, ist aber, dass die numerische Lösung bei nicht zu kleinen Schrittweiten ebenfalls abklingt.

Das einfachste Verfahren, dessen numerische Lösung der Testgleichung (32) bei annehmbaren Schrittweiten das Abklingverhalten der Lösung der Anfangswertaufgabe reproduziert, ist das **implizite Euler Verfahren**

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

Mit ihm erhält man für die Testgleichung $y_{n+1} = y_n + h\lambda y_{n+1}$, d.h.

$$y_n = \Big(\frac{1}{1 - h\lambda}\Big)^n y_0 \to 0 \quad \text{für } n \to \infty$$

für jede Schrittweite $h > 0$. Man geht also einen linearen Schritt mit der Steigung weiter, die in dem Richtungsfeld der Differentialgleichung dort herrscht, wo man hinkommt (Abbildung 4 zeigt 6 Schritte des impliziten Euler Verfahrens für die Anfangswertaufgabe (31) mit der Schrittweite $h = 1$ für $\lambda = -5$).

Der Preis, den man für dieses verbesserte Stabilitätsverhalten zu zahlen hat, ist, dass man im allgemeinen Fall in jedem Schritt ein nichtlineares Gleichungssystem

$$F(y_{n+1}) = y_{n+1} - y_n - hf(x_{n+1}, y_{n+1}) = 0$$

zu lösen hat. Dies kann man z.B. mit dem Newton Verfahren mit dem Startwert y_n tun.

Bemerkung 16 Die Testgleichung (32) ist aussagekräftig für allgemeinere Systeme, denn ist die Matrix $\boldsymbol{A} \in \mathbb{R}^{(n,n)}$ in dem linearen System

$$\boldsymbol{y}' = \boldsymbol{A}\boldsymbol{y} + \boldsymbol{g} \tag{33}$$

diagonalisierbar und gilt

$$\boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X} = \boldsymbol{\Lambda} = \operatorname{diag}\{\lambda_1, \ldots, \lambda_n\}$$

mit einer regulären Matrix $\boldsymbol{X}$, so erhält man mit der Variablentransformation $\boldsymbol{z} := \boldsymbol{X}^{-1}\boldsymbol{y}$

$$\boldsymbol{z}' = \boldsymbol{X}^{-1}\boldsymbol{y}' = \boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X}\boldsymbol{X}^{-1}\boldsymbol{y} + \boldsymbol{X}^{-1}\boldsymbol{g} = \boldsymbol{\Lambda}\boldsymbol{z} + \boldsymbol{X}^{-1}\boldsymbol{g} =: \boldsymbol{\Lambda}\boldsymbol{z} + \tilde{\boldsymbol{g}},$$

d.h. das entkoppelte System

$$z_j' = \lambda_j z_j + \tilde{g}_j, \quad j = 1, \ldots, n. \tag{34}$$

Wendet man auf das System (33) ein Mehrschrittverfahren

$$\sum_{j=0}^{m} \alpha_j \boldsymbol{y}^{k-j} + h \sum_{j=0}^{m} \beta_j (\boldsymbol{A}\boldsymbol{y}^{k-j} + \boldsymbol{g}^{k-j}) = \boldsymbol{0}$$

an, so ist dieses mit $\boldsymbol{z}^i := \boldsymbol{X}^{-1}\boldsymbol{y}^i$ äquivalent zu

$$\sum_{j=0}^{m} \alpha_j \boldsymbol{z}^{k-j} + h \sum_{j=0}^{m} \beta_j (\boldsymbol{\Lambda}\boldsymbol{z}^{k-j} + \tilde{\boldsymbol{g}}^{k-j}) = \boldsymbol{0},$$

d.h. zu dem Mehrschrittverfahren

$$\sum_{j=0}^{m} \alpha_j z_{i,k-j} + h\lambda_i \sum_{j=0}^{m} \beta_j (z_{i,k-j} + \tilde{g}_{i,k-j}) = 0, \quad i = 1, \ldots, n,$$

für die skalaren Gleichungen (34). □

Das Abklingverhalten des Mehrschrittverfahrens für das System (33) wird also bestimmt durch das Abklingverhalten für die skalaren Gleichungen (32) für die Eigenwerte $\lambda = \lambda_j$ der Matrix $\boldsymbol{A}$. Da diese nicht notwendig reell sind, auch wenn $\boldsymbol{A}$ nur reelle Einträge besitzt, müssen wir die Testgleichung (32) für $\lambda \in \mathbb{C}$ mit negativem Realteil untersuchen.

Wendet man das explizite Runge–Kutta Verfahren (18) auf das System (33) an, so ist dieses wieder äquivalent der Anwendung auf die skalaren Gleichungen. Das Abklingverhalten der numerischen Lösung wird auch hier bestimmt durch das Abklingverhalten für die Testgleichung (32). Man erhält für diese

$$\begin{aligned} k_1 &:= \lambda y_n \\ k_j &:= \lambda\Big(y_n + h \sum_{\ell=1}^{j-1} \beta_{j\ell}\, k_\ell\Big), \quad j = 2, \ldots, s, \\ y_{n+1} &:= y_n + h \sum_{j=1}^{s} \gamma_j\, k_j. \end{aligned} \tag{35}$$

Setzt man die k_j nacheinander ein, so folgt

$$y_{n+1} = R(h\lambda) y_n$$

mit

$$R(z) = 1 + z \sum_{j=1}^{s} \gamma_s + z^2 \sum_{j=1}^{s} \sum_{k=1}^{j-1} \gamma_j \beta_{jk} + z^3 \sum_{j=1}^{s} \sum_{k,\ell=1}^{j-1} \gamma_j \beta_{jk} \beta_{k\ell} + \ldots \in \Pi_s,$$

und die Folge $\{y_n\}$ ist offenbar beschränkt, wenn $|R(z)| \leq 1$ gilt. Beachten Sie, dass diese Bedingung nur von $z := h\lambda$ abhängt.

Wir definieren

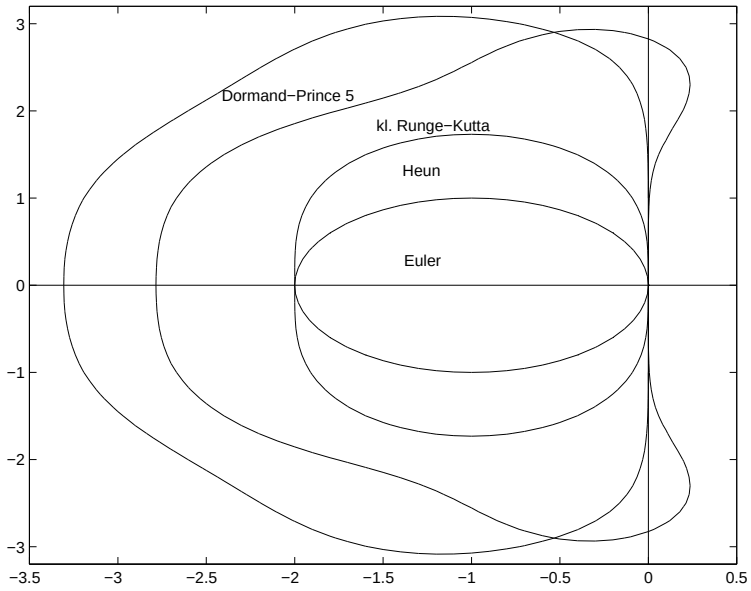

Abbildung 5: Stabilitätsgebiete von Runge–Kutta Verfahren

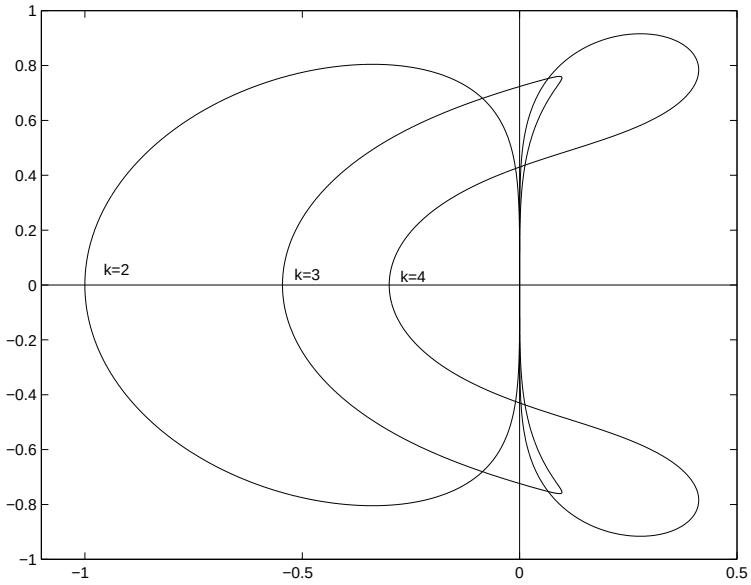

Abbildung 6: Stabilitätsgebiete von Adams–Bashforth Verfahren

Definition 17 *Das* **Stabilitätsgebiet** $S \subset \mathbb{C}$ *eines Verfahrens ist die Menge aller* $z := h\lambda \in \mathbb{C}$*, so dass für alle Startwerte die erzeugte Folge* $\{y_n\}$ *mit der Schrittweite* h *für die Testgleichung (32) beschränkt ist.*

Das Stabilitätsgebiet eines Verfahrens hat die folgende Bedeutung. Will man ein lineares Differentialgleichungssystem $\boldsymbol{y}' = \boldsymbol{A}\boldsymbol{y}$ lösen und besitzt die Matrix $\boldsymbol{A}$ die Eigenwerte λ_j, $j = 1, \ldots, n$, mit $\operatorname{Re}\lambda_j < 0$, so kann man das System nur dann stabil mit diesem Verfahren lösen, wenn die Schrittweite h so klein gewählt ist, dass die Zahlen $h\lambda_j$ für alle $j = 1, \ldots, n$ in dem Stabilitätsgebiet S des Verfahrens liegen.

Abbildung 5 enthält die Stabilitätsgebiete einiger Runge-Kutta Verfahren und Abbildung 6 und Abbildung 7 die Stabilitätsgebiete der ersten Adams–Bashforth und Adams–Moulton Verfahren. Die Gebiete sind alle beschränkt, so dass man bei der Integration von linearen Systemen mit abklingenden Lösungen erhebliche Einschränkungen an die Schrittweiten rechnen muss.

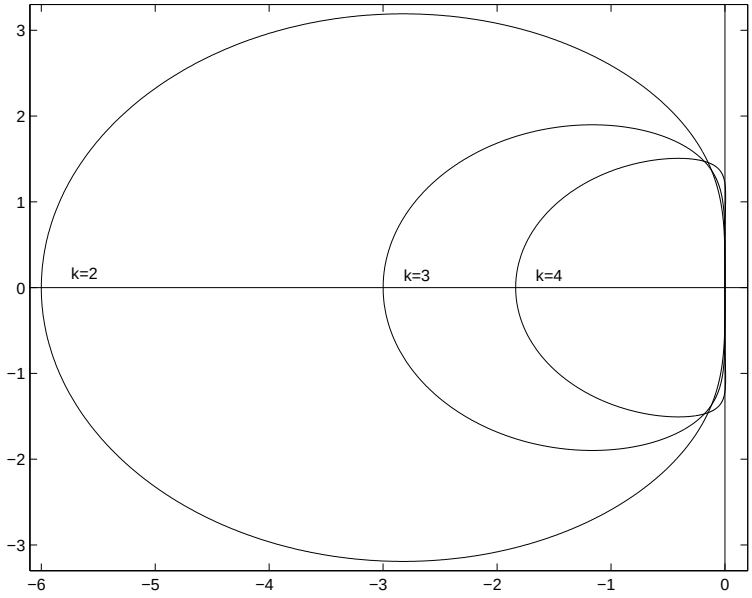

Abbildung 7: Stabilitätsgebiete von Adams–Moulton Verfahren

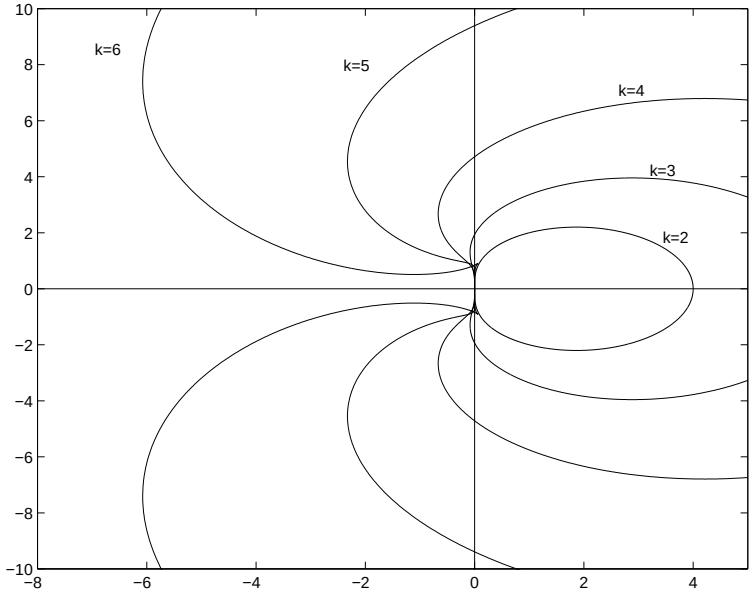

Abbildung 8: Stabilitätsgebiete von BDF-Verfahren

Für die BDF-Verfahren ist dies nicht der Fall. Ihre Stabilitätsgebiete sind in Abbildung 8 gegeben.

Vom Standpunkt der Stabilität sind die BDF-Formeln also den Adams-Verfahren vorzuziehen.

Wünschenswert ist für ein Verfahren, dass sein Stabilitätsgebiet die linke Halbebene umfasst.

Definition 18 *Ein Verfahren zur Lösung von Anfangswertaufgaben heißt* **A–stabil**, *wenn gilt*

$$S \supset \mathbb{C}_- := \{z \in \mathbb{C} : \operatorname{Re} z \leq 0\}.$$

Das implizite Euler Verfahren ist A-stabil (hat aber nur die Konverzordnung 1), nicht aber die expliziten Runge–Kutta, die Adams–Bashforth und die Adams–Moulton Verfahren.

Ein weiteres A-stabiles Verfahren ist die Trapez Regel

$$y_{n+1} = y_n + 0.5h(f_{n+1} + f_n), \tag{36}$$

denn wendet man dieses Verfahren auf die Testgleichung an, so erhält man

$$y_{n+1} = y_n + 0.5h(\lambda y_{n+1} + \lambda y_n) \quad \Longleftrightarrow \quad y_{n+1} = \frac{h\lambda - (-2)}{h\lambda - 2} y_n,$$

und damit

$$S_{\text{Trapez Regel}} = \Big\{ z \in \mathbb{C} \,:\, \Big|\frac{z+2}{z-2}\Big| \leq 1 \Big\} = C_-.$$

Der nächste Satz von Dahlquist zeigt, dass man keine besseren A–stabilen Mehrschrittverfahren finden kann. Einen Beweis findet man in **Hairer, Wanner** [3], p. 247 ff.

Satz 19 (Dahlquist) *(i) Explizite Mehrschrittverfahren sind niemals A-stabil.*

(ii) Die Ordnung eines A–stabilen impliziten Mehrschrittverfahrens ist höchstens 2.

(iii) Die Trapez Regel (36) ist das A-stabile Verfahren der Ordnung 2 mit der kleinsten Fehlerkonstanten.

A–stabile Verfahren (oder wenigstens Verfahren mit einem größeren Stabilitätsgebiet) kann man als Verallgemeinerung der expliziten Runge–Kutta Verfahren erhalten. Bei diesen verwenden wir zur Berechnung von k_j nur die Werte von y_n und k_i, $i = 1, \ldots, j-1$.

Definition 20 *Es seien $\alpha_i, \gamma_i \in \mathbb{R}$, $\beta_{ij} \in \mathbb{R}$, $i, j = 1, \ldots, s$, gegeben. Dann heißt das Verfahren*

$$k_i = f\Big(x_n + \alpha_i h, y_n + h \sum_{j=1}^{s} \beta_{ij} k_j\Big), \; i = 1, \ldots, s, \tag{37}$$

$$y_{n+1} = y_n + h \sum_{i=1}^{s} \gamma_i k_i \tag{38}$$

Runge–Kutta Verfahren *mit s Stufen.*

Gilt $\beta_{ij} = 0$ für $i \leq j$, so handelt es sich um ein explizites Runge–Kutta Verfahren, gilt $\beta_{ij} \neq 0$ für ein $i \leq j$, so nennen wir das Runge–Kutta Verfahren **implizit**.

Beispiel 21 Das implizite Euler Verfahren

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

ist offenbar ein einstufiges implizites Runge–Kutta Verfahren. Ein weiteres naheliegendes Verfahren ist die **implizite Mittelpunktregel**

$$\begin{aligned} k_1 &= f\Big(x_n + \frac{h}{2}, y_n + \frac{1}{2}hk_1\Big) \\ y_{n+1} &= y_n + hk_1, \end{aligned} \tag{39}$$

die offenbar ebenfalls ein einstufiges implizites Runge–Kutta Verfahren ist.

Die Trapez Regel

$$y_{n+1} = y_n + \frac{1}{2}h\Big(f(x_n, y_n) + f(x_{n+1}, y_{n+1})\Big)$$

kann als zweistufiges implizites Runge–Kutta Verfahren aufgefasst werden:

$$\begin{aligned} k_1 &= f(x_n, y_n), \\ k_2 &= f\Big(x_n + h, y_n + h(\frac{1}{2}k_1 + \frac{1}{2}k_2)\Big), \\ y_{n+1} &= y_n + h\Big(\frac{1}{2}k_1 + \frac{1}{2}k_2\Big). \end{aligned}$$

□

Wie die expliziten Runge–Kutta Verfahren stellt man auch die impliziten Verfahren übersichtlich in einem Tableau dar:

$$\begin{array}{c|cccc} \alpha_1 & \beta_{11} & \beta_{12} & \dots & \beta_{1s} \\ \alpha_2 & \beta_{21} & \beta_{22} & \dots & \beta_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_s & \beta_{s1} & \beta_{s2} & \dots & \beta_{ss} \\ \hline & \gamma_1 & \gamma_2 & \dots & \gamma_s. \end{array}$$

Hiermit erhalten die Verfahren aus Beispiel 21 die Gestalt

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array} \qquad \begin{array}{c|c} 0.5 & 0.5 \\ \hline & 1 \end{array} \qquad \begin{array}{c|cc} 0 & 1 & 0 \\ 1 & 0.5 & 0.5 \\ \hline & 0.5 & 0.5 \end{array}$$

impliziter Euler Mittelpunktregel Trapez Regel.

Nachteil der impliziten Runge–Kutta Verfahren ist, dass die k_j nicht nacheinander berechnet werden können, sondern dass in jedem Schritt ein (i.a. nichtlineares) Gleichungssystem von $s \cdot N$ Gleichungen in den $s \cdot N$ Unbekannten $k_1, \dots, k_s$ gelöst werden muss, wobei N die Dimension des Differentialgleichungssystems (1) bezeichnet. Eine naheliegende Frage ist, unter welchen Bedingungen das System (eindeutig) lösbar ist.

Satz 22 *Es sei $f : [a,b] \times \mathbb{R}^N \to \mathbb{R}^N$ stetig und erfülle eine Lipschitz Bedingung bzgl. des zweiten Arguments mit der Lipschitz Konstante L. Erfüllt die Schrittweite $h > 0$ die Bedingung*

$$h < \frac{1}{L \max\limits_{i=1,\dots,s} \alpha_i}, \tag{40}$$

so ist das Gleichungssystem

$$k_i = f\Big(x_n + \alpha_i h, y_n + h \sum_{j=1}^{s} \beta_{ij} k_j\Big), \ i = 1, \dots, s, \tag{41}$$

eindeutig lösbar.

Erfüllt die rechte Seite f nur eine Lipschitzbedingung in einer Umgebung von (x_n, y_n), so bleibt die Aussage des Satzes immer noch richtig, wobei die Schrittweite eventuell verkleinert werden muss, um zu sichern, dass das Argument von f in dieser Umgebung bleibt.

Aus dem Fixpunktsatz für kontrahierende Abbildungen folgt zugleich, dass die Fixpunktiteration gegen die Lösung $(k_i)_{i=1,\dots,s}$ von (41) konvergiert, wenn die Schrittweite h die Bedingung (40) erfüllt. Für steife Probleme ist diese Aussage aber wertlos, da hierfür i.a. die Lipschitzkonstante recht groß sein wird. Bei steifen Systemen muss das Gleichungssystem (41) immer mit dem Newton Verfahren oder einer verwandten Methode gelöst werden.

Zur Bestimmung expliziter Runge–Kutta Verfahren hat man die Lösung eines nichtlinearen Gleichungssystems zu bestimmen. Implizite Runge–Kutta verfahren (auch hoher Ordnung) kann man mit Hilfe des folgenden Satzes konstruieren:

Satz 23 *Die Quadraturformel*

$$\int_{x_n}^{x_n+h} f(x)\,dx = h\sum_{j=1}^{s}\gamma_j f(x_n+\alpha_j h) + O(h^p) \tag{42}$$

besitze die Ordnung p. Dann hat das zugehörige implizite Runge–Kutta Verfahren mit den Parametern α_j, γ_j und

$$\beta_{ij} = \int_0^{\alpha_i}\prod_{\substack{k=1\\k\neq j}}^{s}\frac{t-\alpha_k}{\alpha_j-\alpha_k}\,dt \quad i,j=1,\ldots,s,$$

die Konsistenzordnung p.

Beweis. Deuflhard, Bornemann [1], p. 244 ff.

Nach diesem Ergebnis ist klar, welche Ordnung für implizite Runge–Kutta Verfahren man durch Kollokation maximal erreichen kann und wie man diese erreicht.

Sind die α_j die Knoten der Gaußschen Quadraturformel der Ordnung 2s, d.h. die Nullstellen des s–ten (geschifteten) Legendre Polynoms

$$\frac{d^s}{dx^s}\Big(x^s(1-x)^s\Big),$$

und sind γ_j die zugehörigen Gewichte, so heißt das implizite Runge–Kutta Verfahren aus Satz 23 **s–stufiges Gauß Verfahren**. Das s-stufige Gauß Verfahren hat die Ordnung $2s$.

Es kann keine impliziten Runge–Kutta Verfahren der Stufe s mit höherer Ordnung als $2s$ geben, denn sonst hätte man zugleich durch (42) eine Quadraturformel mit s Knoten und einer höheren Ordnung als $2s$ gefunden.

Man kann zeigen, dass das Gauß Verfahren der Ordnung $2s$ A–stabil ist (vgl. **Hairer, Wanner** [3], p. 72).

Das einstufige Gauß Verfahren ist die implizite Mittelpunktregel, das zweistufige Gauß Verfahren wird durch das Tableau

$$\begin{array}{c|cc}
\frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6}\\
\frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4}\\
\hline
 & \frac{1}{2} & \frac{1}{2}
\end{array}$$

gegeben und das dreistufige Verfahren durch

$$\begin{array}{c|ccc}
\frac{1}{2}-\frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9}-\frac{\sqrt{15}}{15} & \frac{5}{36}-\frac{\sqrt{15}}{30}\\
\frac{1}{2} & \frac{5}{36}+\frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36}-\frac{\sqrt{15}}{24}\\
\frac{1}{2}+\frac{\sqrt{15}}{10} & \frac{5}{36}+\frac{\sqrt{15}}{30} & \frac{2}{9}+\frac{\sqrt{15}}{15} & \frac{5}{36}\\
\hline
 & \frac{5}{18} & \frac{4}{9} & \frac{5}{18}
\end{array}$$

Wir haben bereits bemerkt, dass die nichtlinearen Gleichungssysteme, die in den impliziten Runge–Kutta Verfahren auftreten, nicht durch sukzessive Iteration gelöst werden können, sondern dass eine Linearisierung verwendet werden muss. Um dies zu umgehen, wurde von Rosenbrock 1962 vorgeschlagen, nur den linearen Anteil der rechten Seite mit einem

impliziten Verfahren zu behandeln und den Rest mit einem expliziten Verfahren. Wir erläutern das Vorgehen für das autonome System

$$\boldsymbol{y}' = \boldsymbol{f}(\boldsymbol{y}). \tag{43}$$

Dies bedeutet keine Einschränkung, denn das nichtautonome System

$$\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y})$$

kann man durch Hinzunahme der Differentialgleichung

$$x' = 1$$

in ein autonomes System

$$\begin{array}{rcl} \boldsymbol{y}' & = & \boldsymbol{f}(x, \boldsymbol{y}) \\ x' & = & 1 \end{array} \tag{44}$$

transformieren.

(43) schreiben wir mit $\boldsymbol{J} := \boldsymbol{f}'(\boldsymbol{y})$ als

$$\boldsymbol{y}' = \boldsymbol{J}\boldsymbol{y}(x) + (\boldsymbol{f}(\boldsymbol{y}(x)) - \boldsymbol{J}\boldsymbol{y}(x)) \tag{45}$$

und berechnen hierfür für $i = 1, \ldots, s$ die "Steigungen"

$$\boldsymbol{k}_i = \boldsymbol{J}\Big(\boldsymbol{y} + h\sum_{j=1}^{i} \tilde{\beta}_{ij}\boldsymbol{k}_j\Big) + \Big(\boldsymbol{f}(\boldsymbol{y} + h\sum_{j=1}^{i-1} \beta_{ij}\boldsymbol{k}_j) - \boldsymbol{J}(\boldsymbol{y} + h\sum_{j=1}^{i-1} \beta_{ij}\boldsymbol{k}_j)\Big). \tag{46}$$

Beachten Sie, dass das aus (46) zu berechnende $\boldsymbol{k}_i$ auf der rechten Seite nur linear auftritt und nur die bereits bekannten $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_{i-1}$ auch im Argument der nichtlinearen Funktion $\boldsymbol{f}$. Man kann also $\boldsymbol{k}_i$ aus einem linearen Gleichungssystem berechnen.

Definition 24 *Ein Schritt eines* **linear impliziten Runge–Kutta Verfahrens** *oder* **Rosenbrock Verfahrens** *mit s Stufen für das autonome System (43) besteht aus den folgenden Schritten*

$$\begin{array}{lrcl} (i) & \boldsymbol{J} & = & \boldsymbol{f}'(\boldsymbol{y}_n) \\ (ii) & (\boldsymbol{I} - h\tilde{\beta}_{ii}\boldsymbol{J})\boldsymbol{k}_i & = & h\sum\limits_{j=1}^{i-1}(\tilde{\beta}_{ij} - \beta_{ij})\boldsymbol{J}\boldsymbol{k}_j + \boldsymbol{f}(\boldsymbol{y}_n + h\sum\limits_{j=1}^{i-1}\beta_{ij}\boldsymbol{k}_j), \\ & & & i = 1, \ldots, s \\ (iii) & \boldsymbol{y}_{n+1} & = & \boldsymbol{y}_n + h\sum\limits_{j=1}^{s}\gamma_j\boldsymbol{k}_j \end{array} \tag{47}$$

In jedem Schritt sind s lineare Gleichungssysteme zu lösen. Eine weitere Vereinfachung tritt ein, wenn man $\tilde{\beta}_{ii} = \tilde{\beta}$ fordert. Man benötigt dann nur eine LR–Zerlegung zur Lösung der s Systeme.

Bemerkung 25 Wendet man das linear implizite Verfahren (47) auf das "autonomisierte" System (44) an, so kann man die zu der Variablen x gehörige Gleichung explizit ausrechnen. Man erhält die folgende Form des Verfahrens

$$\begin{array}{rcl} \boldsymbol{J} & = & \dfrac{\partial}{\partial \boldsymbol{y}}\boldsymbol{f}(x_n, \boldsymbol{y}_n), \\ (\boldsymbol{I} - h\tilde{\beta}_{ii}\boldsymbol{J})\boldsymbol{k}_i & = & \boldsymbol{f}(x_n + \alpha_i h, \boldsymbol{y}_n + h\sum\limits_{j=1}^{i-1}\beta_{ij}\boldsymbol{k}_j) + \delta_i h\dfrac{\partial}{\partial x}\boldsymbol{f}(x_n, \boldsymbol{y}_n) \\ & & +h\boldsymbol{J}\sum\limits_{j=1}^{i-1}(\tilde{\beta}_{ij} - \beta_{ij})\boldsymbol{k}_j \;\; i = 1, \ldots, s, \\ \boldsymbol{y}_{n+1} & = & \boldsymbol{y}_n + h\sum\limits_{j=1}^{s}\gamma_j\boldsymbol{k}_j \end{array} \tag{48}$$

Dabei ist

$$\alpha_i := \sum_{j=1}^{i-1} \beta_{ij},\ \delta_i := \tilde{\beta}_{ii} \sum_{j=1}^{i-1} (\tilde{\beta}_{ij} - \beta_{ij}).$$

□

Beispiel 26 Das einfachste linear implizite Runge–Kutta Verfahren ist das **linear implizite Euler Verfahren**

$$\begin{aligned} \Big(\boldsymbol{I} - h\frac{\partial}{\partial \boldsymbol{y}} \boldsymbol{f}(x_n, \boldsymbol{y}_n)\Big)\boldsymbol{k}_1 &= \boldsymbol{f}(x, \boldsymbol{y}_n) + h\frac{\partial}{\partial x} \boldsymbol{f}(x_n, \boldsymbol{y}_n) \\ \boldsymbol{y}_n &= \boldsymbol{y}_n + h\boldsymbol{k}_1 \end{aligned} \tag{49}$$

Dieses besitzt dieselbe Stabilitätsfunktion wie das implizite Euler Verfahren und ist daher ebenfalls A–stabil. Die Konsistenzordnung ist ebenfalls 1. □

Rosenbrock Verfahren höherer Ordnung wurden von Kaps und Rentrop (1979), Shampine (1982) und van Veldhuizen (1984) konstruiert. Kaps und Ostermann (1989) konstruierten eingebettete linear implizite Runge–Kutta Verfahren. Sie können ähnlich leicht implementiert werden wie explizite Runge–Kutta Verfahren. Den Code ROS4 findet man in

http://www.unige.ch/math/folks/hairer/

Eine Abschwächung der A-Stabilität ist die $A(\alpha)$-Stabilität:

Definition 27 *Ein Verfahren heißt $A(\alpha)$–stabil, wenn sein Stabilitätsgebiet den Sektor*

$$S_\alpha := \{z \in \mathbb{C} : |arg(-z)| \leq \alpha\} \tag{50}$$

enthält.

Bei vielen Anwendungen weiß man, dass die Eigenwerte der Linearisierung in einem festen Sektor der Gestalt (50) liegen, so dass man stabiles Verhalten der numerischen Lösung für alle Schrittweiten erwarten kann.

Die BDF-Formeln für $k \leq 6$ sind $A(\alpha)$–stabil (vgl. Abbildung 8). Für den Öffnungswinkel α gilt

k	1	2	3	4	5	6
α	90°	90°	86.0°	73.3°	51.8°	17.8°

In MATLAB sind 2 Methoden zur Lösung steifer Systeme implementiert. Das Programm ode15s verwendet BDF-Formeln oder **NDF-Verfahren** (numerical differentiation formulas) der Ordnung $k \in \{1, 2, 3, 4, 5\}$. NDF sind Modifikationen der BDF-Methoden, die ebenfalls $A(\alpha)$-stabil sind. Sie besitzen eine etwas größere Genauigkeit als die BDF-Methoden, wobei der Öffnungswinkel α des maximalen im Stabilitätsgebiet enthaltenen Sektors aber nur wenig verkleinert wird. Ihre Konstruktion ist in **Shampine und Reichel** [5] beschrieben.

Das Programm ode23s verwendet ein Rosenbrock Verfahren der Ordnung 2, wobei der Fehler mit einer Modifikation der Ordnung 3 geschätzt wird. Es ist geeignet, wenn die Genauigkeitsansprüche nicht zu hoch sind.

Es gibt viele (auch einander widersprechende) Definitionen der Steifheit. Häufig wird gesagt, dass ein Problem steif ist, wenn es verschieden schnell abklingende Lösungen besitzt,

z.B. weil die Jacobi Matrix der rechten Seite Eigenwerte mit sehr unterschiedlichen (negativen) Realteilen besitzt.

Dies trifft jedoch nicht den Kern. Will man das schnelle Abklingen von Lösungskomponenten darstellen, so ist man gezwungen, die Lösungen mit sehr kleinen Schrittweiten zu approximieren, und hierzu kann man dann ein explizites Verfahren verwenden, denn diese sind in der Regel billiger als implizite Verfahren. Will man dagegen (wie in Beispiel 5) eine sich langsam ändernde Lösung verfolgen, was eigentlich mit großen Schrittweiten möglich sein sollte, und wird ein explizites Verfahren durch sehr schnell abklingende Lösungsanteile zu sehr kleinen Schrittweiten gezwungen, so nennen wir das Problem **steif**.

Aufgabe 6 *(a) Bestimmen Sie die Lösung der Anfangswertaufgabe*

$$y'' = 100y,\ y(0) = 1,\ y'(0) = -10, \quad 0 \le x \le 5.$$

(b) Lösen Sie die Anfangswertaufgabe mit den Lösern ode23, ode45, ode113, ode23s und ode15s, die in MATLAB implementiert sind. Hat die Beobachtung irgendetwas mit Steifheit zu tun?

Aufgabe 7 *Die Anfangswertaufgabe*

$$\begin{aligned} y_1' &= -0.01y_1 + 10^3 y_2 y_3 & y_1(0) &= 1 \\ y_2' &= 0.01y_1 - 10^3 y_2 y_3 - 10^7 y_2^2 & y_2(0) &= 0 \\ y_3' &= 10^7 y_2^2 & y_3(0) &= 0 \end{aligned}$$

beschreibt eine chemische Reaktion von 3 Reaktanden.

(a) Bestimmen Sie die Lösung im Intervall $[0, 10^6]$ *mit einem der steifen Löser der ODE suite.*

(b) Bestimmen Sie die Lösung mit den Runge–Kutta Lösern ODE23 und ODE45 im Intervall $[0, 5]$*. Vergleichen Sie die zweite Komponente im Intervall* $[0, 5]$ *mit der Lösung aus Teil (a).*

(c) Bestimmen Sie die Lösung im Intervall $[0, 10^6]$ *mit dem impliziten Euler Verfahren (wie kann man hierzu die in der MATLAB ODE Suite vorhandenen Löser nutzen?)*

5 Abschließende Bemerkungen

Einer Anfangswertaufgabe sieht man nicht unmittelbar an, ob ihre Lösung steif ist. Es gibt einige Aufgabenklassen, bei denen man weiß, dass steife Lösungen zu erwarten sind wie z.B. Gleichungen, die chemische Reaktionen mit sehr unterschiedlichen Reaktionsgeschwindigkeiten beschreiben, Systeme, die sich mit der Linienmethode aus parabolischen oder hyperbolischen Anfangsrandwertaufgaben ergeben, oder singulär gestörte Probleme wie die van der Pol Gleichung mit sehr großem Parameter. In diesem Fall wird man sofort steife Löser verwenden.

Liegen keine guten Gründe dafür vor, dass eine steife Lösung zu erwarten ist, wird man zuerst versuchen, das gegebene Problem mit einem nicht-steifen Löser zu behandeln, denn explizite (eigebettete) Runge–Kutta Verfahren oder Mehrschrittverfahren vom Adams Typ sind wesentlich billiger als steife Löser. Bei steifen Lösern hat man ja in jedem Schritt ein nichtlineares Gleichungssystem zu lösen und hierzu die Jacobimatrix der rechten Seite oder eine Näherung davon in jedem Schritt bestimmen. Beobachtet man, dass der Lösungsprozess nur sehr langsam voranschreitet, wird man zu einem steifen Löser wechseln.

Runge–Kutta Verfahren ermöglichen eine einfache Schrittweitensteuerung, haben aber den Nachteil gegenüber den Adams Verfahren, dass in jedem Schritt die rechte Seite an mehreren Stellen ausgewertet werden muss (für das Verfahren von Dormand und Prince der Ordnung 5 an 6 Stellen). Beim Prädiktor–Korrektor Verfahren kann man hohe Ordnungen mit 2 (im Fall PECE) oder 3 (im Fall PECECE) Auswertungen erreichen. Man wird daher ein Mehrschrittverfahren verwenden, wenn die Auswertung der rechten Seite der Differentialgleichung sehr teuer ist. In beiden Fällen wird man Verfahren hoher Ordnung nur dann verwenden, wenn die rechte Seite der Differentialgleichung sehr glatt ist. Man verwendet ja den Taylorschen Satz, um Methoden hoher Konsistenzordnung zu entwickeln.

Eine Regel für die Auswahl steifer Löser ist nicht so einfach zu formulieren. Einen Anhaltspunkt geben die Stabilitätsgebiete der Verfahren. BDF Formeln sind $A(\alpha)$–stabil, wobei für große Ordnungen die Winkel α klein sind. Wenn man weiß, dass die Eigenwerte der Linearisierung der rechten Seite in der Nähe der negativen reellen Achse liegen (bei Linienmethoden für parabolische Aufgaben liegen sie sogar auf der negativen reellen Achse), so wird man BDF Formeln wählen. Weiß man, dass Eigenwerte der Jacobimatrix näher an der imaginären Achse als an der negativen reellen Achse liegen (bei Linienmethoden für hyperbolische Probleme liegen sie auf der imaginären Achse), so wird man Rosenbrockmethoden verwenden.

Literatur

[1] P. Deuflhard, F. Bornemann: Numerische Mathematik II. Integration gewöhnlicher Differentialgleichungen. De Gruyter, Berlin 1994

[2] E. Hairer, S.P. Nørsett, G. Wanner: Solving Ordinary Differential Equations I. Nonstiff Problems. 2. Auflage. Springer Verlag, Berlin 1993

[3] E. Hairer, G. Wanner: Solving Ordinary Differential Equations II. Stiff and Differential–Algebraic Problems. 2. Auflage. Springer Verlag, Berlin 1996

[4] L.F. Shampine: Numerical Solution of Ordinary Differential Equations. Chapman & Hall, New York 1994

[5] L.F. Shampine, M.W. Reichel: The MATLAB ODE suite. SIAM J. Sci. Comput. 18, 1 – 22 (1997)

[6] SIMULINK Dynamic System Simulation Software. User's Guide. The Mathworks, Natick 1992

[7] H. Voß: Numerische Behandlung von Differentialgleichungen. Skript, TU Hamburg-Harburg 1998