(5.13) Implementierung von RKV

$y \in \mathbb{R}^d$
s Stufenzahl

Bei impliziten RKV müssen i.a. nichtlineare Gleichungssysteme

$$Y_i = y_0 + h \sum_{j=1}^{s} a_{ij} f(t_0 + c_j h, Y_j), \qquad i = 1, \ldots, s$$

der Dimension ds gelöst werden. Mit den Lösungen Y_i berechnet man dann explizit

$$y_1 = y_0 + h \sum_{i=1}^{s} b_i f(t_0 + c_i h, Y_i).$$

Hierzu werden s zusätzliche Funktionsauswertungen benötigt.

Um den Einfluss von Rundungsfehlern zu reduzieren löst man die Runge-Kutta-Gleichungen nicht nach Y_i, sondern nach $Z_i = Y_i - y_0$, $i = 1, \ldots, s$ auf. Definieren wir

$$Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_s \end{bmatrix}, \qquad f(Y) = \begin{bmatrix} f(t_0 + c_1 h, Y_1) \\ \vdots \\ f(t_0 + c_s h, Y_s) \end{bmatrix},$$

$Y = y$

und entsprechend Z, dann sind die Runge-Kutta-Gleichungen äquivalent zu

$$Y = \mathbb{1} \otimes y_0 + h(\mathcal{A} \otimes I) f(Y).$$

$e = \mathbb{1} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}$

Hierbei bezeichnet $A \otimes B$ das Kronecker-Produkt zweier Matrizen: Für $A \in \mathbb{C}^{m,n}$ und $B \in \mathbb{C}^{p,q}$ ist

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \cdots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix} \in \mathbb{C}^{mp,nq}.$$

Mit obiger Notation ist $Z = Y - \mathbb{1} \otimes y_0$, also erhalten wir

$$Z = h(\mathcal{A} \otimes I) f(\mathbb{1} \otimes y_0 + Z),$$
$$y_1 = y_0 + h(b^T \otimes I) f(\mathbb{1} \otimes y_0 + Z).$$

Bei der Berechnung von y_1 können Fehler in Z durch die große Lipschitz-Konstante von f verstärkt werden. Falls $\mathcal{A}$ invertierbar ist, kann man dieses Problem durch eine weitere Umformung umgehen. Aus

$$(\mathcal{A}^{-1} \otimes I) Z = h f(\mathbb{1} \otimes y_0 + Z)$$

folgt nämlich

$$y_1 = y_0 + (b^T \otimes I)(\mathcal{A}^{-1} \otimes I) Z$$
$$=: y_0 + (d^T \otimes I) Z, \qquad \text{mit } d^T = b^T \mathcal{A}^{-1}.$$

Zusätzliche Funktionsauswertungen sind hier nicht mehr nötig. Im Spezialfall, dass $a_{si} = b_i$ für alle i gilt, etwa bei Radau-Verfahren, ist $d^T = e_s^T$ und damit $y_1 = y_0 + Z_s = Y_s$.

Lösung der nichtlinearen Gleichungssysteme für Z_i, $i = 1, \ldots, s$ mit vereinfachten Newton-Verfahren
Da $||Z_i|| = O(h)$ wählt man als Startwert $Z_i^{(0)} = 0$.
Definieren wir

$$F_i(Z) = Z_i - h \sum_{j=1}^{s} a_{ij} f(t_0 + c_j h, y_0 + Z_j), \qquad i = 1, \ldots, s$$

und $F(Z) = (F_i(Z))_{i=1}^s$, so ist das nichtlineare Gleichungssystem $F(Z) = 0$ zu lösen. Es gilt

$$F'(Z) = I_{sd} - h \begin{bmatrix} a_{11} f_y(t_0 + c_1 h, y_0 + Z_1) & \cdots & a_{1s} f_y(t_0 + c_s h, y_0 + Z_s) \\ \vdots & & \vdots \\ a_{s1} f_y(t_0 + c_1 h, y_0 + Z_1) & \cdots & a_{ss} f_y(t_0 + c_s h, y_0 + Z_s). \end{bmatrix}.$$

Im vereinfachten Newton-Verfahren ersetzen wir alle Jacobi-Matrizen $f_y(t_0 + c_j h, y_0 + Z_j)$ durch eine Approximation $J \approx f_y(t_0, y_0)$:

$$F'(Z) \approx I_{sd} - h\mathcal{A} \otimes J.$$

Iterationsvorschrift des vereinfachten Newton-Verfahrens:

$$\begin{aligned}(I_{sd} - h\mathcal{A} \otimes J)\Delta Z^{(k)} &= -F(Z^{(k)}) \quad \leftarrow \text{Löse lineares Gleichungssystem } ds \times ds \\ &= -Z^{(k)} + h(\mathcal{A} \otimes I) f(\mathbb{1} \otimes y_0 + Z^{(k)}), \\ Z^{(k+1)} &= Z^{(k)} + \Delta Z^{(k)}\end{aligned}$$

Multiplikation mit $(h\mathcal{A} \otimes I)^{-1} = h^{-1}(\mathcal{A}^{-1} \otimes I)$ ergibt

$$(h^{-1}(\mathcal{A}^{-1} \otimes I) - I \otimes J)\Delta Z^{(k)} = -h^{-1}(\mathcal{A}^{-1} \otimes I)Z^{(k)} + f(\mathbb{1} \otimes y_0 + Z^{(k)})$$

Im vereinfachten Newton-Verfahren ersetzen wir alle Jacobi-Matrizen $f_y(t_0 + c_j h, y_0 + Z_j)$ durch eine Approximation $J \approx f_y(t_0, y_0)$:

$$F'(Z) \approx I_{sd} - h\mathcal{A} \otimes J.$$

Iterationsvorschrift des vereinfachten Newton-Verfahrens:

$$\begin{aligned}(I_{sd} - h\mathcal{A} \otimes J)\Delta Z^{(k)} &= -F(Z^{(k)}) \\ &= -Z^{(k)} + h(\mathcal{A} \otimes I) f(\mathbb{1} \otimes y_0 + Z^{(k)}), \\ Z^{(k+1)} &= Z^{(k)} + \Delta Z^{(k)}\end{aligned}$$

Multiplikation mit $(h\mathcal{A} \otimes I)^{-1} = h^{-1}(\mathcal{A}^{-1} \otimes I)$ ergibt

$$\rightarrow \quad (h^{-1}(\mathcal{A}^{-1} \otimes I) - I_s \otimes J)\Delta Z^{(k)} = -h^{-1}(\mathcal{A}^{-1} \otimes I)Z^{(k)} + f(\mathbb{1} \otimes y_0 + Z^{(k)})$$

$T \Lambda T^{-1} \qquad T I_s T^{-1}$

Die direkte Lösung des linearen Gleichungssystems mit Gauß-Elimination, oder mit LU-Zerlegung der Koeffizientenmatrix, kostet $\frac{1}{3}(sd)^3$ Operationen.
Effizienter ist es, $\mathcal{A}^{-1}$ zu diagonalisieren:

$$T^{-1}\mathcal{A}^{-1}T = \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_s).$$

Die Diagonalisierung hängt nur vom Verfahren ab, nicht von der Differentialgleichung. Sie muss nur ein einziges Mal berechnet werden.

Mit einfachen Rechenregeln für Kronecker-Produkte erhält man für die transformierten Variablen $W = (T^{-1} \otimes I_d)Z$

$$(h^{-1}\Lambda \otimes I_d - I_s \otimes J)\Delta W^{(k)} = -h^{-1}(\Lambda \otimes I)W^{(k)} + (T^{-1} \otimes I)f(\mathbb{1} \otimes y_0 + (T \otimes I)W^{(k)}).$$

Die Koeffizientenmatrix dieses linearen Gleichungssystems ist blockdiagonal. Die s Systeme der Dimension d sind also entkoppelt. Damit man nicht für alle s Eigenwerte λ_i eine neue LU-Zerlegung berechnen muss transformiert man J zuerst auf obere Hessenberg-Form, $H = Q^H J Q$, mit einer unitären Matrix Q. Der Aufwand hierfür beträgt $\frac{2}{3}d^3$ Operationen. Die LU-Zerlegung von $h^{-1}\lambda_i I - H$ kann dann mit $O(d^2)$ Operationen berechnet werden.

$s=3$

$$\begin{bmatrix} h^{-1}\lambda_1 - J & 0 & 0 \\ 0 & h^{-1}\lambda_2 - J & 0 \\ 0 & 0 & h^{-1}\lambda_3 - J \end{bmatrix} \begin{bmatrix} \Delta W_1^{(k)} \\ \Delta W_2^{(k)} \\ \Delta W_3^{(k)} \end{bmatrix}$$

$$\left(h^{-1}\lambda_i I_d - J\right)\Delta W_i^{(k)} \qquad i = 1..s$$

$$H = \begin{bmatrix} * & \cdots & * \\ \ddots & \ddots & \vdots \\ 0 & * & * \end{bmatrix}$$

Algorithm 1 Vereinfachtes Newton-Verfahren

	Aufwand
$W^{(0)} = (T^{-1} \otimes I)Z^{(0)}$	$s^2 d$
Transformiere J auf Hessenberg-Form: $H = Q^H J Q$	$2d^3/3$
Berechne LU-Zerlegungen von $h^{-1}\lambda_i I - H$ für $i = 1, \ldots, s$	sd^2
$k = 0$	
while $(k < k_{\max})$ **do**	
Auswertung $f(t_0 + c_i h, y_0 + Z_i^{(k)})$, $i = 1, \ldots, s$	s f-Ausw.
$r = (T^{-1} \otimes I)f(\mathbb{1} \otimes y_0 + Z^{(k)})$	$s^2 d$
$r = -h^{-1}(\Lambda \otimes I)W^{(k)} + r$	sd
Löse $(h^{-1}\lambda_i I - H)(Q\Delta W_i^{(k)}) = Q r_i$ mit LU-Zerlegung, $i = 1, \ldots, s$	$2sd^2$
Berechne $\|\|\Delta W^{(k)}\|\|$ für Konvergenztest	sd
$W^{(k+1)} = W^{(k)} + (Q^H \otimes I)((Q \otimes I)\Delta W^{(k)})$	$sd + sd^2$
$Z^{(k+1)} = (T \otimes I)W^{(k+1)}$	$s^2 d$
$k = k + 1$	
end while	

Konvergenz oder Divergenz des vereinfachten Newton-Verfahrens liefert Hinweise zur Wahl der Schrittweite und zur erneuten Auswertung der Jacobi-Matrix.
Man reduziert die Schrittweite falls innerhalb der vorgegeben maximalen Anzahl von Newton-Schritten keine Konvergenz eintritt.

Zur Schrittweitensteuerung werden eingebettete Verfahren verwendet, vgl. Abschnitt (3.23). Hat $\mathcal{A}^{-1}$ einen reellen Eigenwert (also insbesondere bei allen Verfahren mit ungerader Stufenzahl), können eingebettete Verfahren so gewählt werden, dass keine zusätzliche LU-Zerlegung berechnet werden muss. Wichtig ist, dass das eingebettete Verfahren auch für steife Differentialgleichungen geeignet ist. Die Fehlerschätzung angewendet auf die Testgleichung sollte sich wie der exakte Fehler verhalten.

Details zur Implementierung finden Sie in [Hairer Wanner Abschnitt IV.8]

(5.14) Differentialgleichungen mit einseitiger Lipschitzbedingung

Voraussetzung für diesen Abschnitt

$\dot{y} = f(t,y)$ $\quad f: \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ stetig diffbar

und erfülle die einseitige Lipschitzbedingung

$$(EL) \qquad \langle f(t,x) - f(t,z),\ x - z \rangle \leq \ell \, \|x - z\|^2 \qquad \forall\, x, z \in \mathbb{R}^d$$

$\langle \cdot, \cdot \rangle$ euklidisches Skalarprodukt

$\|\cdot\| = \langle \cdot, \cdot \rangle^{1/2}$ euklidische Norm

Beispiel 1

$\dot{y} = -1000\, y \quad \rightsquigarrow \quad \ell = -1000$ oder größer

Aber "die" Lipschitzkonstante ist + 1000 oder größer $\quad \|f(t,y) - f(t,z)\| \leq L \, \|y - z\|$

Bemerkung Falls f Lipschitzstetig mit Lipschitzkonstante L ist, ist (EL) mit $\ell = L$ erfüllt. Für das kleinstmögliche ℓ in (EL) kann $\ell \ll L$ gelten

Satz 2 (vgl. Satz (2.5))

Sei für f (EL) erfüllt und seien $y, z : [t_0, T] \to \mathbb{R}$ Lösungen

$$\begin{cases} \dot{y} = f(t, y) \\ y(t_0) = y_0 \end{cases} \quad \text{bzw} \quad \begin{cases} \dot{z}(t) = f(t, z) \\ z(t_0) = z_0 \end{cases}$$

Dann gilt für $t \in [t_0, T]$

$$\| y(t) - z(t) \| \leq e^{\ell (t - t_0)} \| y_0 - z_0 \|$$

Beweis: $\frac{d}{dt} \| y(t) - z(t) \|^2 = \frac{d}{dt} \langle y(t) - z(t), y(t) - z(t) \rangle$

$$= 2 \langle \dot{y}(t) - \dot{z}(t), y(t) - z(t) \rangle = 2 \langle f(t, y(t)) - f(t, z(t)), y(t) - z(t) \rangle$$
$$\leq 2 \ell \| y(t) - z(t) \|^2$$

Für $y_0 \neq z_0$ ist, da die Lösung eindeutig ist $y(t) \neq z(t)$ für alle t und damit $\| y(t) - z(t) \|^2 > 0$

$$\Rightarrow \quad \frac{d/dt \, \| y(t) - z(t) \|^2}{\| y(t) - z(t) \|^2} = \frac{d}{dt} \log \left(\| y(t) - z(t) \|^2 \right) \leq 2\ell$$

$\int_{t_0}^{t}$

$$\Rightarrow \quad \log \left(\| y(t) - z(t) \|^2 \right) - \log \left(\| y_0 - z_0 \|^2 \right) \leq 2\ell (t - t_0)$$

$e^{\cdot}$

$$\Rightarrow \quad \| y(t) - z(t) \|^2 \leq e^{2\ell (t - t_0)} \| y_0 - z_0 \|^2$$

$\sqrt{\cdot}$

$$\Rightarrow \quad \| y(t) - z(t) \| \leq e^{\ell (t - t_0)} \| y_0 - z_0 \|$$

□

Satz 3 (vgl. Satz (3.9))

Unter der Voraussetzung (EL) und falls $h \cdot \ell < 1$ gilt für den Fehler des impliziten Eulerverfahrens

$$\| y_n - y(t_n) \| \leq m h \quad \text{für alle} \quad t_n \in t_0 + nh \in [t_0, T]$$

mit $$m = \frac{1}{\ell} \left(e^{\ell (T-t_0)/(1-h\ell)} - 1 \right) \frac{1}{2} \max_{t \in [t_0, T]} \{ \ddot{y}(t) \}$$

Beweis i) lokales Fehler

Starte auf exakter Lösung und betrachte Fehler nach einem Schritt

$$y(t_n + h) = y(t_n) + h \dot{y}(t_n + h) + \frac{h^2}{2} \ddot{y}(\tau_n) \quad \text{für } \tau_n \in [t_n, t_n + h]$$

$$= y(t_n) + h f(t_n + h, y(t_n + h)) + d_{n+1}$$

wobei $$\| d_{n+1} \| \leq \frac{h^2}{2} \max_{\tau \in [t_0, T]} \| \ddot{y}(\tau) \|$$

Sei z_{n+1} Approximation des impl. Eulerverfahrens ausgehend von $y(t_n)$ nach einem Schritt

$$z_{n+1} = y(t_n) + h f(t_n + h, z_{n+1})$$

$$\rightarrow \| y(t_n + h) - z_{n+1} \|^2$$

$$= \langle y(t_n) + h f(t_n + h, y(t_n + h)) + d_{n+1} - \left(y(t_n) + h f(t_n + h, z_{n+1}) \right), y(t_n + h) - z_{n+1} \rangle$$

$$\leq h \ell \| y(t_n + h) - z_{n+1} \|^2 + c h^2 \| y(t_n + h) - z_{n+1} \|$$

für $$c = \frac{1}{2} \max_{\tau \in [t_0, T]} \| \ddot{y}(\tau) \|$$

$$\Rightarrow \| y(t_n + h) - z_{n+1} \| \leq \frac{c h^2}{1 - h\ell} \qquad (1 - h\ell > 0 \text{ nach Voraussetzung})$$

ii) Fehlerfortpflanzung

Prüfungstermine Mo 11. 8 Mo 12. 10
Mo 31. 8