D-ITET, D-MATL
Dr. R. Käppeli
P. Bansal

Serie 10

1. *Zu Einfaches adaptives Runge-Kutta-Fehlberg Verfahren*

In dieser Aufgabe wollen wir wie in Aufgabe 3 aus Serie 9 etwas mit adaptiver Schrittweitensteuerung experimentieren. Zur lokalen Fehlerschätzung verwenden wir die eingebettete Runge-Kutta-Verfahren welche in Paragraph III.2.2 der Vorlesungs-Notizen beschrieben ist.

Das adaptive Runge-Kutta-Fehlberg Verfahren, das auch RKF45 genannt wird, ist ein sehr bekanntes adaptives Verfahren basierend auf eingebetteten Runge-Kutta Verfahren der Ordnung 4 und 5. Sein Butcher Schema lautet

0						
1/4	1/4					
3/8	3/32	9/32				
12/13	1932/2197	-7200/2197	7296/2197			
1	439/216	-8	3680/513	-845/4104		
1/2	-8/27	2	-3544/2565	1859/4104	-11/40	
	16/135	0	6656/12825	28561/56430	-9/50	2/55
	25/216	0	1408/2565	2197/4104	-1/5	0

Die erste Zeile von b_i-Koeffizienten entspricht dem Verfahren fünfter Ordnung, die zweite dem Verfahren vierter Ordnung.

a) Implementieren Sie ein adaptives Verfahren auch basierend auf dem einfachen Algorithmus aus Serie 9 Aufgabe 3.
Hinweis: Arbeiten Sie im Template `RKF45Simple.m`.

b) Testen Sie, ob die Verfahren getrennt voneinander Konvergenzresultate vierter bzw. fünfter Ordnung produzieren.

c) Wenden Sie das Verfahren auf das Van der Popol Problem an, welches in **Aufgabe 3.b) aus Serie 9** definiert wurde. Verwenden Sie als absolute und relative Toleranzen 100`atol` = `rtol` = 10^{-3}. Was beobachten Sie?

Bitte wenden!

2. *Adaptive Schrittweitensteuerung*

a) Modifizieren Sie die MATLAB Funktionen `RKF45.m` und `adaptHeun.m` von das letzte Aufgabe und Serie 9 Aufgabe 3 gemäss Algorithmus auf Seite 12 von Kapitel III. Beachten Sie folgende Punkte:

- Falls die Schrittweite `h` kleiner als eine gegebene Toleranz `hmin` wird, soll der Algorithmus enden.
- Für den Algorithmus basierend auf dem Heun-Verfahren und der Intervall-Halbierungs-Methode, benutzen Sie den Fehlerschätzer $\hat{\epsilon}_{j+1}$ gegeben auf Seite 7 von Kapitel III.

b) Wiederholen Sie **Aufgabe 3.b) aus Serie 9** mit den modifizierten Algorithmen. Was beobachten Sie? Erklären Sie warum die modifizierte Algorithmen besser sind.

3. *Explizites und Implizites Euler-Verfahren*

Für das Anfangswertproblem $\dot{y}(t) = f(t, y(t))$ ist ein Schritt des impliziten Euler-Verfahrens durch

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

definiert.

a) Betrachten Sie folgendes AWP

$$\dot{y}(t) = -\lambda y(t), \qquad y(t_0) = y_0.$$

Führen Sie (analytisch) einen Schritt (mit Schrittweite h) mit dem expliziten und impliziten Euler-Verfahren aus.

b) Betrachten Sie folgendes AWP

$$\dot{y}(t) = -t(y(t))^2, \qquad y(t_0) = y_0 > 0,$$

Führen Sie (analytisch) einen Schritt (mit Schrittweite h) mit dem expliziten und impliziten Euler-Verfahren aus.

c) Lösen Sie das AWP

$$\dot{y}(t) = -20y(t), \qquad y(0) = 1,$$

jeweils mit dem expliziten und impliziten Euler-Verfahren mit Schrittweiten $h = 2^{-1}, \ldots, 2^{-8}$ und Endzeit $T = 1$. Plotten Sie die Lösung für beide Verfahren und jede Schrittweite. Was beobachten Sie?

Siehe nächstes Blatt!

4. *Mehrschrittverfahren: Das 2-Schrittverfahren von Adams-Bashforth*

Die uns nun vertrauten Einschrittverfahren berechnen die Approximation der Lösung y_{j+1} zur Zeit t_{j+1} allein mittels der Approximation der Lösung y_j zur Zeit t_j. Dagegen benutzen sog. *Mehrschrittverfahren* zur Berechnung von y_{j+1} zusätzlich auch bekannte Approximationen der Lösung zu vorhergehenden Zeiten t_{j-1}, t_{j-2}, ... Der Einfachheit halber betrachten wir im folgenden nur eine konstante Schrittweite h.

Ausgangspunkt ist wiederum dir zur (skalaren) Differentialgleichung äquivalente Integralgleichung

$$y(t_{j+1}) = y(t_j) + \int_{t_j}^{t_{j+1}} f(\tau, y(\tau))\, d\tau \tag{1}$$

Die Idee ist nun das Integral mittels Interpolation zu approximieren. Hierzu seien Approximationen der Lösung zur Zeit t_j sowie zu den $m-1$ vorhergehenden Zeiten $t_{j-1} := t_j - h$, $\ldots$, $t_{j+1-m} := t_j - (m-1)h$ bekannt, d.h. wir kennen y_j, $\ldots$, y_{j+1-m} und damit auch $f_j := f(t_j, y_j)$, $\ldots$, $f_{j+1-m} := f(t_{j+1-m}, y_{j+1-m})$. Daraus bilden wir das eindeutig bestimmte Interpolationspolynom zu den m Stützpunkten (t_j, y_j), $\ldots$, (t_{j+1-m}, y_{j+1-m}):

$$P_{m-1}(t) = \sum_{k=1}^{m} f_{j+1-k}\, L^m_{j+1-k}(t)$$

wobei

$$L^m_{j+1-k}(t) = \prod_{\substack{l=1 \\ l \neq k}}^{m} \frac{t - t_{j+1-l}}{t_{j+1-k} - t_{j+1-l}}$$

die Lagrange-Polynome sind. Wir verwenden nun $P_m(t)$ zur Approximation des Integrals in (1) und erhalten somit folgende Ausdruck

$$y_{j+1} = y_j + \int_{t_j}^{t_{j+1}} P_{m-1}(\tau)\, d\tau.$$

Dies ist das sog. m*-Schrittverfahren von Adams-Bashforth* (ABm).

a) Bauen Sie nach obigem Rezept das 2-Schrittverfahren von Adams-Bashforth (AB2).

b) Implementieren Sie AB2 in MATLAB. Auf was für eine Komplikation stossen Sie bei der Implementierung.
Hinweis: Verwenden Sie das Heun Verfahren.

c) Messen Sie die Konvergenzordnung von AB2 an folgendem AWP:

$$\dot{y}(t) = -2y(t),\ y(0) = 5.$$

Bitte wenden!

d) Begründen Sie die gemessene Konvergenzordnung indem Sie die Konsistenzordnung von AB2 bestimmen.
Hinweis: Das Prinzip ist dasselbe wie bei Einschrittverfahren.

Abgabe: Bis Freitag, den 17.05.2019.