D-ITET, D-MATL
Dr. R. Käppeli
R.Hohl

Serie 9

1. *Konsistenz Expliziter Runge-Kutta Verfahren*

Wir betrachten ein s-stufiges explizites Runge-Kutta Verfahren definiert durch das folgende Butcher-Schema

$$\begin{array}{c|cccc} c_1 & 0 & & & 0 \\ c_2 & a_{21} & 0 & & \\ \vdots & \vdots & & \ddots & \\ c_s & a_{s1} & & a_{s,s-1} & 0 \\ \hline & b_1 & & \dots & b_s \end{array}$$

Zeigen Sie, dass die Bedingung $\sum_{i=1}^{s} b_i = 1$ notwendig ist für die Konsistenz des Verfahrens.

Hinweis: Zeigen Sie zuerst, dass für den Schritt $y_0 \to y_1$ für die Stufen gilt: $k_i(h) = f(t_0, y_0) + \mathcal{O}(h)$, $i = 1, \dots, s$.

2. *Toleranz Variieren*

Betrachten Sie folgendes AWP

$$\dot{y}(t) = \lambda \left(y(t) - \frac{t^2}{1+t^2} \right) + \frac{2t}{(1+t^2)^2}, \quad y(0) = 0, \tag{1}$$

mit $\lambda = 10$ auf dem Zeitintervall $[0, 5]$.

a) Verifizieren Sie, dass

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

das AWP (1) löst.

Bitte wenden!

b) Lösen Sie das AWP mit `ode45` und folgenden absoluten/relativen Toleranzen:

$$(\tau_{abs}, \tau_{rel}) = (10^{-6}, 10^{-3}),\ (10^{-6}, 10^{-6}),\ (10^{-8}, 10^{-8}),\ (10^{-10}, 10^{-10}).$$

Was beobachten Sie?

Hinweis: Verwenden Sie das MATLAB-Template `tolVar.m`.

c) Sei nun $\lambda = -10$. Geben Sie die exakte Lösung das AWP an.

d) Wiederholen Sie **b)** mit $\lambda = -10$.

e) Erklären Sie das beobachtete Verhalten in **b)** und **c)**.

3. *Zu Einfaches Adaptives Heun-Verfahren*

In dieser Aufgabe wollen wir etwas mit adaptiver Schrittweitensteuerung experimentieren. Als Basis-Verfahren soll das Heun-Verfahren

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

verwendet werden. Zur lokalen Fehlerschätzung verwenden wir die Schrittweitenhalbierungs Methode welche in Paragraph III.2.1 der Vorlesungs-Notizen beschrieben ist. Bei dieser Methode berechnet man ausgehend von der Lösung im j-ten Schritt $\mathbf{y}_j$:

(i) einen Schritt mit Schrittweite h : $\mathbf{y}_j \to \mathbf{y}_{j+1}$,

(ii) zwei Schritte mit Schrittweite $h/2$: $\mathbf{y}_j \to \hat{\mathbf{y}}_{j+1}$.

Damit berechnet man die Fehlerschätzer ε_{j+1} und $\hat{\varepsilon}_{j+1}$ (s. III.2.1). Man überlegt sich dann folgenden einfachen Algorithmus:

```
function [t,y] = adaptHeunSimple(f,t0,T,y0,h0,atol,rtol)

  while (t_j < T)

    % Gegeben y_j und h_j berechne y_{j+1} und ŷ_{j+1}

    % Berechne lokalen Fehlerschatzer ε̂_{j+1}

    if ( ε̂_{j+1} < atol + ||y_j|| rtol ) % akzeptiere Zeitschritt!
      t_{j+1} = t_j + h_j
      y_{j+1} = ŷ_{j+1}
    else % verwerfe Zeitschritt und halbiere Schrittweite
```

Siehe nächstes Blatt!

$h_j = h_j/2$
end

end

Hier ist `f` die rechte Seite der Diff.-Gleich., `t0` die Anfangszeit, `T` die Endzeit, `y0` der Anfangswert, `h0` die Anfangs-Schrittweite, `atol` und `rtol` die abolute und relative Toleranz. Der obige Algorithmus verwendet (willkürlich!) das Toleranz-Kriterium TK4 aus der Vorlesung (s. Seite 4 der Notizen für weitere Möglichkeiten).

a) Implementieren Sie in MATLAB das oben beschriebene einfache adaptive Heun-Verfahren.
Hinweis: Arbeiten Sie im Template `adaptHeunSimple.m`

b) Lösen Sie mit Ihrem `adaptHeunSimple.m` aus **a)** die Van der Pol-Gleichung[1]

$$\ddot{y}(t) = 8(1 - y(t)^2)\dot{y}(t) - y(t)$$

für $t \in [0, 30]$ mit den Anfangswerten

$$y(0) = 2 \quad , \quad \dot{y}(0) = 0.$$

Verwenden Sie als absolute und relative Toleranzen `atol` = `rtol` = 10^{-5} und als Anfangs-Schrittweite `h0` = 1. Plotten Sie die Lösung und die Schrittweite. Was beobachten Sie?
Hinweis: Um die Schrittweite zu berechnen könnte der Befehl `h = diff(t)` nützlich sein.

c) Welche Schwächen hat der obige adaptive Algorithmus?

Abgabe: Bis Freitag, den 08.05.2020.

[1] Aufgestellt vom niederländischen Elektroingenieur und Physiker Balthasar van der Pol zur Beschreibung von Oszillatoren während er bei Philips tätig war.