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

Serie 10

1. *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
      h_j = h_j/2
    end

  end
```

Bitte wenden!

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?

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

In dieser Aufgabe wollen wir wie in Aufgabe 1 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

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

Siehe nächstes Blatt!

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 Aufgabe 1.
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 **1.b)** definiert wurde. Verwenden Sie als absolute und relative Toleranzen 100`atol` = `rtol` = 10^{-3}. Was beobachten Sie?

3. *Adaptive Schrittweitensteuerung*

a) Modifizieren Sie die MATLAB Funktionen `RKF45.m` und `adaptHeun.m` von Aufgaben **1)** und **2)** 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 **1.b)** mit den modifizierten Algorithmen. Was beobachten Sie? Erklären Sie warum die modifizierte Algorithmen besser sind.

Abgabe: Bis Freitag, den 18.05.2017.