

Serie 9

1. *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 Schrittweithalberungs 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 y_j :

- (i) einen Schritt mit Schrittweite h : $y_j \rightarrow y_{j+1}$,
- (ii) zwei Schritte mit Schrittweite $h/2$: $y_j \rightarrow \hat{y}_{j+1}$.

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

    % Berechne lokalen Fehlerschatzer \hat{\varepsilon}_{j+1}

    if ( \hat{\varepsilon}_{j+1} < atol + \|y_j\| rtol ) % akzeptiere Zeitschritt!
        t_{j+1} = t_j + h_j
        y_{j+1} = \hat{y}_{j+1}
    else % verwerfe Zeitschritt und halbiere Schrittweite
        h_j = h_j/2
    end

end

end
```

Bitte wenden!

Hier ist f die rechte Seite der Diff.-Gleich., t_0 die Anfangszeit, T die Endzeit, y_0 der Anfangswert, h_0 die Anfangs-Schrittweite, $atol$ und $rtol$ die absolute und relative Toleranz (Toleranz-Kriterium TK4 aus der Vorlesung). Dieses adaptive Verfahren ist bereits in `adaptHeunSimple.m` implementiert.

Wir lösen mit diesem Verfahren die Van der Pol-Gleichung¹

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

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

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

Wir verwenden absolute und relative Toleranzen $atol = rtol = 10^{-5}$ und als Anfangs-Schrittweite $h_0 = 1$.

- Das Skript `vanDerPol_adaptHeun.m` plottet die Lösung und die Schrittweite für das obige AWP. Was beobachten Sie?
- Welche Schwächen hat der obige adaptive Algorithmus?
- Verbessern Sie `adaptHeunSimple.m` in dem Sie den Algorithmus auf Seite 12 von Kapitel III der Vorlesung implementieren.
Bauen Sie auch eine Sicherheit ein, damit der Algorithmus endet falls eine untere Schranke $h_{min} = 10 * eps$ für die Schrittweite unterschritten wird.

2. *Adaptives Runge-Kutta-Fehlberg Verfahren*

In dieser Aufgabe wollen wir wie in der vorherigen Aufgabe etwas mit adaptiver Schrittweitensteuerung experimentieren. Zur lokalen Fehlerschätzung verwenden wir nun ein eingebettetes Runge-Kutta-Verfahren (siehe III.2.2 Vorlesung).

Das adaptive Runge-Kutta-Fehlberg Verfahren (auch RKF45 genannt) ist ein bekanntes adaptives Verfahren basierend auf zwei (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

¹Aufgestellt 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. Dieses Verfahren mit dem (zu) einfachen Algorithmus aus der vorherigen Aufgabe ist bereit in `RKF45Simple.m` implementiert.

- a) Überprüfen Sie graphisch das die Verfahren getrennt voneinander Konvergenzresultate vierter bzw. fünfter Ordnung produzieren. Führen Sie hierzu das Skript `KonvTestRKF45.m` aus. (Dieses Skript berechnet einfach das Konvergenzverhalten der Verfahren für ein skalares AWP.)
- b) Das Skript `vanDerPol_RKF45.m` plottet die Lösung und die Schrittweite für das Van der Pol AWP aus der vorherigen Aufgabe. Offensichtlich weist der zu einfache Algorithmus immer noch dieselben Probleme auf. Verbessern Sie `RKF45Simple.m` in dem Sie den Algorithmus auf Seite 12 von Kapitel III der Vorlesung implementieren.
Bauen Sie auch eine Sicherheit ein, damit der Algorithmus endet falls eine untere Schranke `hmin = 10*eps` für die Schrittweite unterschritten wird.

3. 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}, \dots . Der Einfachheit halber betrachten wir im folgenden nur eine konstante Schrittweite h .

Ausgangspunkt ist wiederum die zur (skalaren) Differentialgleichung äquivalente Integralgleichung

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

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 $t_{j-1} := t_j - h, \dots, t_{j+1-m} := t_j - (m-1)h$ bekannt, d.h. wir kennen y_j, \dots, y_{j+1-m} und damit auch $f_j := f(t_j, y_j), \dots, f_{j+1-m} := f(t_{j+1-m}, y_{j+1-m})$. Daraus bilden wir das eindeutig bestimmte Interpolationspolynom zu den m Stützpunkten $(t_j, y_j), \dots, (t_{j+1-m}, y_{j+1-m})$:

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

wobei

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

Bitte wenden!

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 (AB_m)*.

- 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: Arbeiten Sie im Template AB2.m und verwenden Sie z.B. das Heun Verfahren.
- c) Messen Sie die Konvergenzordnung von AB₂ an folgendem AWP:

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

Hinweis: Das Template KonvTestAB2.m implementiert bereits alles.

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

Abgabe: Online bis **Freitag** den 12.05.2023 unter `sam-up.math.ethz.ch`.