Universität Basel
Mathematisches Institut

Dr. Julia Schweitzer
Christian Stohrer
Fabian Lukas Müller
13.10.2010

Numerik gewöhnlicher Differentialgleichungen (HS 2010)

Allgemeines

Das Projekt besteht aus zwei aufeinander aufbauenden Teilen, die je einen Kreditpunkt wert sind. Die Projekte zählen als erfolgreich bearbeiten, wenn die Programme funktionieren, ein kurzer Bericht (ca. 3 Seiten für beide Teile zusammen, im Wesentlichen Dokumentation über Programmieraufgaben und Ergebnisse) geschrieben ist, und die Ergebnisse mit dem Dozenten oder den Assistenten besprochen wurden. Es ist möglich, nur den ersten Teil des Projekts zu bearbeiten.

Adaptive Implementierung von expliziten Runge-Kutta-Verfahren

Zur Numerik gehört nicht nur die Entwicklung und theoretische Untersuchung von Verfahren, sondern auch die effiziente Implementierung der Verfahren. Bei expliziten Runge-Kutta-Verfahren

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

$$y_1 = y_0 + h \sum_{i=1}^{s} b_i f(t_0 + c_i h, Y_i) \tag{2}$$

angewandt auf die Differentialgleichung

$$y'(t) = f(t, y(t)) \ , \quad y(t_0) = y_0 \tag{3}$$

steckt der Hauptrechenaufwand in den Auswertungen von f, da diese meist komplizierte Funktionen sind, die durch die unterschiedlichen, konkreten Probleme gegeben sind. Der Rest des Verfahrens beschränkt sich auf Multiplikationen von Vektoren mit Zahlen und Additionen von Vektoren.

Wenn man also den Gesamtaufwand betrachtet, ist dieser durch die Anzahl der Schritte bestimmt, wobei der Aufwand pro Schritt im Wesentlichen durch die Anzahl von Auswertungen von f gegeben ist.

Eine effiziente Implementierung eines Runge-Kutta-Verfahrens sollte also sowohl die Anzahl der Schritte, als auch die Anzahl der Funktionsauswertungen pro Schritt so gering wie möglich halten, um die vom Benutzer vorgegebenen Anforderungen an die Lösung zu gewährleisten. Die typischen Anforderungen sind Genauigkeit der Lösung und Zeitpunkte, an denen die Lösung benötigt wird. Beide werden in diesem Projekt behandelt.

Beispiel: Arenstorf-Orbit

Als Beispiel kann man die Bewegung eines Satelliten im Gravitationsfeld von Erde und Mond betrachten. Dabei handelt es sich um ein Drei-Körper-Problem, dass nicht exakt lösbar ist.

Die Gesamtmasse der drei Körper sei M und die Masse von Erde und Mond zusammen sei $\tilde{\mu}M$. Die Masse des Satelliten ist gegenüber der Masse von Erde und Mond praktisch vernachlässigbar,

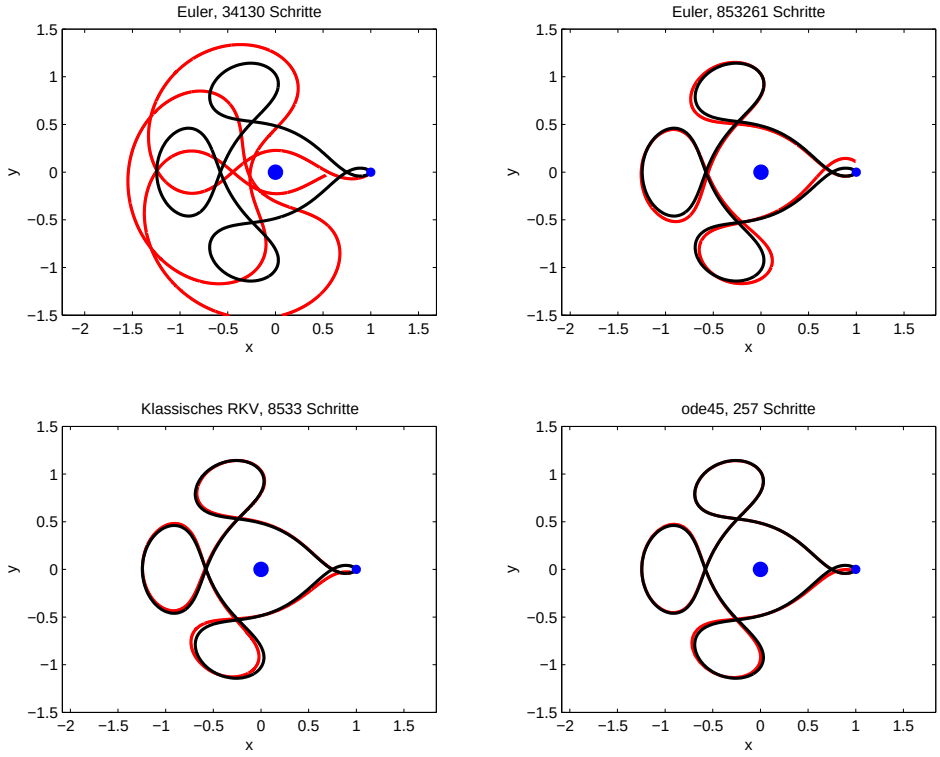

Abbildung 1: Arenstorf-Orbit mit verschiedenen Verfahren: Die schwarze Kurve ist die Referenzlösung gerechnet mit `ode45` und einer sehr feinen Toleranz.

daher gilt $\tilde{\mu} \approx 1$. Die Bahnkurve des Satelliten wird im mitrotierenden Schwerpunktsystem von Erde und Mond dargestellt, d.h. der Schwerpunkt der beiden Körper befindet sich im Ursprung und Erde wowie Mond haben eine konstante Position in diesem System.

Diese Bahnkurve des Satelliten wird Arenstorf-Orbit genannt, nach den von Richard Arenstorf berechneten Bahnen, die im Apollo-Programm der USA eine wichtige Rolle bei der Planung der Reise zum Mond spielten.

Die Gleichungen für die Koordinaten (x, y) des Satelliten im mitrotierendem System sind durch die Gleichungen

$$x'' = x + 2y' - \tilde{\mu}\frac{x+\mu}{D_1} - \mu\frac{x-\tilde{\mu}}{D_2} \qquad D_1 = \left((x+\mu)^2 + y^2\right)^{3/2} \tag{4a}$$

$$y'' = y - 2x' - \tilde{\mu}\frac{y}{D_1} - \mu\frac{y}{D_2} \qquad D_2 = \left((x-\tilde{\mu})^2 + y^2\right)^{3/2} \tag{4b}$$

Für Erde und Mond gilt $\tilde{\mu} = 1 - \mu$ mit $\mu = 0.012277471$. Für gewisse Anfangswerte existieren periodische Bahnen mit Periode T. Ein solches Set ist

$$x(0) = 0.994\ , \qquad x'(0) = 0\ .$$
$$y(0) = 0\ , \qquad y'(0) = -2.0015851063790825\ ,$$

für das sich die Periodendaufer $T = 17.065216560157962558 89$ ergibt. Für die Zeit entspricht eine

Einheit einem Monat und die Länge wird in der mittleren Entfernung von Erde und Mond (ca. 384 000 km) gemessen.

Die Ergebnisse für das explizite Euler-Verfahren und das klassische Runge-Kutta-Verfahren gerechnet mit konstanter Schrittweite sind in Abbildung 1 dargestellt. Ausserdem ist die Lösung berechnet mit dem Matlab-Löser `ode45`, welcher ein explizites Runge-Kutta-Verfahren mit Schrittweitensteuerung ist, gezeigt. Dieser kommt mit sehr viel weniger Schritten aus, als die anderen Verfahren mit konstanter Schrittweite.

Projekt 1: Schrittweitensteuerung mittels eingebetteter Verfahren

(1 Kreditpunkt)

Es sei die Differentialgleichung (3) gegeben und die Lösung sei eine Funktion, die zu unterschiedlichen Zeiten sehr unterschiedliches Verhalten zeigt. Da die Fehlerkonstante von Ableitungen der Lösung abhängt, kann sie an manchen Stellen also sehr gross sein, an anderen wiederum sehr klein. Wenn man nun überall die selbe Schrittweite h benutzt, hat man an den Stellen, wo die Konstante klein ist, sehr kleine, lokale Fehler, da wo die Konstante gross ist, aber viel grössere Fehler. Man braucht also entweder in einem Bereich zu viele Schritte, was den Rechenaufwand in die Höhe treibt, oder man hat eine Lösung, die zu grosse numerische Fehler enthält.

Ziel der Schrittweitensteuerung ist es also, das gesamte Integrationsintervall so zu unterteilen, dass auf allen Teilintervallen der lokale Fehler ungefähr gleich gross und unter einer vorgegebenen Toleranz ist. Da die exakte Lösung und damit der Fehler natürlich nicht bekannt sind, muss man nun Verfahren betrachten, die den Fehler schätzen.

Dazu berechnet man zwei numerische Lösungen; y_1 mit einem Verfahren der Ordnung p und $\hat{y}_1$ mit einem Verfahren der Ordnung $p-1$. Für die beiden Lösungen gilt dann

$$y_1 - y(t_0+h) = \mathcal{O}(h^{p+1}) \ , \quad \hat{y}_1 - y(t_0+h) = Ch^p + \mathcal{O}(h^{p+1}) \ ,$$

wobei $C = C(t_0, y_0)$ nur von dem Intervall $[t_0, t_0+h]$ abhängt. Damit kann man den lokalen Fehler durch

$$e_1 := y_1 - \hat{y}_1 \approx Ch^p$$

abschätzen und eine Approximation an die Konstante

$$C \approx \frac{1}{h^p} e_1$$

herleiten. Hat man eine Toleranz τ vorgegeben, wäre die ideale Schrittweite h_{opt} diejenige, für die $\|Ch_{\text{opt}}^p\| = \tau$ gilt. In diesem Fall hätte man gerade eben die Toleranz erreicht, aber die maximal grosse Schrittweite verwendet. Aus der Approximation der Konstante C lässt sich h_{opt} bestimmen:

$$\|e_1\| \Big(\frac{h_{\text{opt}}}{h}\Big)^p = \tau \quad \Leftrightarrow \quad h_{\text{opt}} = h \sqrt[p]{\frac{\tau}{\|e_1\|}}$$

Aus diesen Überlegungen lässt sich ebenfalls schliessen, ob die aktuelle Näherung gut genug ist, d.h. $\|e_1\| \leq \tau$, oder ob man den Schritt mit einer kleineren Schrittweite wiederholen sollte, falls $\|e_1\| > \tau$. Hat man den Schritt verworfen, rechnet man ihn mit dem neu gegebenen h_{opt}, welches dann kleiner ist, als das letzte h, neu. Wurde der Schritt akzeptiert, rechnet man mit h_{opt} weiter. Hierbei ist zu beachten, dass h_{opt} zu der geschätzten Konstante auf $[t_0, t_0+h]$ gehört. Da aber $C(t_0+h, y_1) = C(t_0, y_0) + \mathcal{O}(h)$ gilt, ist es trotzdem sinnvoll, h_{opt} zu verwenden.

Bis dahin ist die theoretische Vorgehensweise klar. Nun muss man sich überlegen, wie man das praktisch umsetzt. Zunächst wäre da die Berechnung von $\hat{y}_1$. Hier kann man Rechenaufwand sparen, wenn man sogenannte eingebettete Verfahren benutzt, um $\hat{y}_1$ zu berechnen. Wenn das eigentliche Runge-Kutta-Verfahren durch die Koeffizienten a_{ij}, $i,j = 1,\dots s$ und b_i, $i = 1,\dots s$ gegeben ist, behält man die inneren Stufen Y_i vom ursprünglichen Verfahren bei und ändert nur die Formel für y_1 zu

$$\hat{y}_1 = y_0 + h \sum_{i=1}^{s} \hat{b}_i Y_i' \ .$$

Da die inneren Stufen gleich bleiben, kommen bei dieser Vorgehensweise keine zusätzlichen Auswertungen von f hinzu. Bei der Konstruktion dieser Verfahren ist zu beachten, dass das eingebettete Verfahren genau Ordnung $p-1$ hat. Dafür müssen alle Ordnungsbedingungen für Bäume mit höchstens $p-1$ Knoten erfüllt sein. Von den Bedingungen für Bäume mit p Knoten muss aber mindestens eine nicht erfüllt sein. Je mehr davon nicht erfüllt sind, desto aussagekräftiger ist der Schätzer, da das eingebettete Verfahren sonst für bestimmte Differentialgleichungen zufällig eine bessere Ordnung haben könnte.

Des weiteren muss man sich noch über die Toleranz sowie die Norm, in der man den Fehler misst, Gedanken machen. Komponentenweise verlangt man, dass der Fehler

$$|y_1^I - \hat{y}_1^I| \leq sc^I := \tau_A^I + \max\{|y_0^I|, |y_1^I|\}\tau_R^I$$

erfüllt. Für $\tau_A = 0$, $I = 1, \ldots, d$ misst man damit den relativen Fehler, für $\tau_R = 0$, $I = 1, \ldots, d$ misst man den absoluten Fehler. In der Praxis verwendet man häufig eine gemischte Skalierung, bei der beide Toleranzen positiv sind. Damit definiert man die Norm

$$\|e_1\| = \sqrt{\frac{1}{d}\sum_{I=1}^{d}\Big(\frac{e_1^I}{sc^I}\Big)^2}\ . \tag{5}$$

In einem guten Code sollte man vorsichtig mit derartigen Schätzungen umgehen. Daher führt man einige "Sicherheitsfaktoren" ein. Zum Beispiel multipliziert man die optimale Schrittweite noch mit einem Faktor, der etwa als $fac = 0.8, 0.9$ oder $0.25^{1/p}$ gewählt wird, damit der nächste Schritt mit hoher Wahrscheinlichkeit akzeptiert wird. Zudem verbietet man, dass die Schrittweite zu schnell erhöht oder reduziert wird und setzt schliesslich

$$h_{\text{neu}} = h \cdot \min\Big\{facmax, \max\big\{facmin, fac \cdot \frac{1}{\|e_1\|^{1/p}}\big\}\Big\}\ .$$

Mögliche Werte sind z.B. $facmin = 1/3$ und $facmax$ zwischen 1.5 und 5. Nach einem verworfenen Schritt kann man $facmax$ vorrübergehend auf 1 setzen, um eine direkte Erhöhung der Schrittweite zu verhindern.

Aufgabe 1a:
Zeigen Sie, dass mit der Fehlernorm definiert in Gleichung (5) ein Schritt akzeptiert wird, wenn $\|e_1\| \leq 1$ gilt, und dass er verworfen wird, falls $\|e_1\| > 1$ gilt.

Aufgabe 1b:
Das 4-stufige klassische Runge-Kutta-Verfahren kann als 5-stufiges Verfahren mit dem Tableau

$$\begin{array}{c|ccccc}
0 & 0 & & & & \\
\frac{1}{2} & \frac{1}{2} & 0 & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & 0 & & \\
1 & 0 & 0 & 1 & 0 & \\
1 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & 0 \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & 0
\end{array}$$

interpretiert werden. Zeigen Sie, dass für die Koeffizienten

$$\hat{b}_1 = \frac{1}{6}\ , \quad \hat{b}_2 = \frac{1}{3}\ , \quad \hat{b}_3 = \frac{1}{3}\ , \quad \hat{b}_4 = 0\ , \quad \hat{b}_5 = \frac{1}{6}$$

das eingebettete Verfahren Ordnung 3 aber nicht Ordnung 4 hat.

Aufgabe 1c:
Programmieren Sie das Klassische Runge-Kutta-Verfahren mit Schrittweitensteuerung, wobei Sie das eingebettete Verfahren aus Aufgabe 1b verwenden. Ein Template für dieses Programm ist angegeben.

Template 1: Matlab Tempalte für ein Runge-Kutta-Verfahren mit Schrittweitensteuerung.

```
function [t,y] = RKV_SW(f, tspan, y0, Atol, Rtol, h0)
% Eingabe:
% f function handle fuer rechte Seite der DGL
% tspan = [t0 T] Integrationsintervall
% y0 Startwert
% Atol absolute Toleranz
% Rtol relative Toleranz
% h0 Startschrittweite
%
% Ausgabe
% t = [t0, t0 + h0, ... T] enthaelt alle Zeitstellen
% y = [y0, y1, ... yn] enthaelt die numerischen L\"osungen

% Setze Parameter

% Integrationsschleife:
while % noch nicht fertig integriert

  % Berechne neue Loesung

  % Berechne geschaetzten Fehler

  if % Fehler klein genug?
    % Schritt akzeptieren, Daten speichern, neue Schrittweite berechnen

  else
    % Schritt ablehnen, Daten nicht speichern, neue Schrittweite berechnen

  end
  % ist die neue Schrittweite laenger als der Rest des Intervalls?

end
```

Aufgabe 1d:
Wenden Sie Ihr Programm auf die Differentialgleichung (4) für den Arenstorf-Orbit mit den oben angegebenen Anfangsdaten und Parametern an. Schreiben Sie sie dazu in ein System erster Ordnung um.

Benutzen Sie unterschiedliche Toleranzen und plotten Sie die Lösung gegen die Zeit, wobei Sie die Stellen kenntlich machen, an denen die Lösung berechnet wurde. Plotten Sie ausserdem die gewählte Zeitschrittweite semilogarithmisch gegen die Zeit.

Projekt 2: Stetige numerische Lösungen von Runge-Kutta-Verfahren

(1 Kreditpunkt)

Numerische Verfahren zur Lösung von Differentialgleichungen liefern Approximationen an die Lösung zu diskreten Zeitpunkten $t_n = t_{n-1} + h_{n-1}$, wobei die h_i z.B. durch eine Schrittweitensteuerung wie aus Projekt 1 bestimmt werden. Man hat typischerweise keine Näherung an $y(t_{n-1}+\theta h_{n-1})$ für $0 < \theta < 1$. In manchen Anwendungen ist es aber nötig, die Lösung an beliebigen Zwischenstellen zu berechnen. Dies sollte natürlich ohne grossen Mehraufwand, insbesondere ohne weitere Auswertungen der Funktion f geschehen. Daher müssen wir mit dem auskommen, was wir für das Runge-Kutta-Verfahren bereits ausgerechnet haben, nämlich die internen Stufen Y_i und deren Ableitungen Y_i'. Des weiteren kann man für explizite Verfahren noch ohne Mehraufwand y_n und $f(y_n)$ benutzen, da man das für den darauffolgenden Schritt sowieso berechnen muss. Formal definieren wir uns dazu ein Verfahren, wo y_n die letzte interne Stufe wird:

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array} \quad \Rightarrow \quad \begin{array}{c|ccc|c} c_1 & a_{11} & \cdots & a_{1s} & 0 \\ \vdots & \vdots & & \vdots & 0 \\ c_s & a_{s1} & \cdots & a_{ss} & 0 \\ 1 & b_1 & \cdots & b_s & 0 \\ \hline & b_1 & \cdots & b_s & 0 \end{array}$$

Für dieses neue $(s+1)$-stufige Verfahren, das formal die selbe Lösung liefert, wie das ursprüngliche, werden also $c_{s+1} = 1$, $a_{si} = b_i$, $i = 1, \ldots, s$, $a_{i(s+1)} = 0$ für $i = 1, \ldots, s+1$ und $b_{s+1} = 0$ ergänzt.

Die stetige, numerische Lösung wird als

$$y_\theta = y_0 + h \sum_{i=1}^{s^*} b_i(\theta) Y_i' \tag{6}$$

definiert, wobei nun die Gewichte $b_i(\theta)$ Funktionen von θ sind und $s^* = s$ für das ursprüngliche oder $s^* = s+1$ für das erweiterte Verfahren gilt. Die Gewichte sollten so konstruiert werden, dass

$$y_\theta - y(t_0 + \theta h) = \mathcal{O}(h^{p^*+1})$$

gilt, wobei p^* die Ordnung des Fehlers der stetigen Lösung ist. So, wie man Ordnungsbedingungen für die Verfahren mittels Bäumen hergeleitet hat, kann man das hier nun auch tun. Die Taylor-Entwicklung der numerischen Lösung ändert sich nicht, wenn man anstelle der Gewichte b_i die neuen Funktionen $b_i(\theta)$ verwendet. Wenn man aber die exakte Lösung an der Stelle $y(t_0 + \theta h)$ entwickelt, erhält man zusaetzliche Potenzen von θ. Die Ordnungsbedingungen lauten dann

$$\gamma(\tau)\Phi_\theta(\tau) = \theta^{\rho(\tau)}$$

wobei die $\Phi_\theta(\tau)$ analog zu den $\Phi(\tau)$ aus den ursprünglichen Ordnungsbedingungen gebildet werden, nur dass man anstelle der b_i die $b_i(\theta)$ einsetzt,

$$\Phi_\theta(\tau) = \sum_{i=1}^{s^*} b_i(\theta)\Phi_i(\tau) \ .$$

Hierbei enthalten die $\Phi_i(\tau)$ die Summen über die Koeffizienten a_{ij}, die durch das ursprüngliche Verfahren bereits bekannt sind. Als nächstes nehmen wir an, dass die $b_i(\theta)$ Polynome in θ sind,

$$b_i(\theta) = \sum_{j=1}^{p^*} b_{ij}\theta^j \ .$$

Hier ist zu beachten, dass die Polynome keine konstanten Terme enthalten. Das ist sinnvoll, da für $\theta = 0$ die numerische, stetige Lösung $y_\theta = y_0$ erfüllen soll. Wenn man das Polynom in die Ordnungsbedingungen einsetzt, erhält man lineare Gleichungssysteme zur Bestimmung der Koeffizienten b_{ij}.

Es ist möglich, dass die Ordnung p^* der stetigen Runge-Kutta-Lösung kleiner als die des Verfahrens selber ist, ausserdem ist die stetige numerische Lösung im Allgemeinen nicht auf dem gesamten Zeitintervall differenzierbar.

Aufgabe 2a:
Zeigen Sie, dass die Ordnungsbedingungen für die Koeffizienten der stetigen numerischen Lösung auf das Gleichungssystem

$$\begin{pmatrix} 1 & \cdots & 1 \\ \Phi_1(\tau_{21}) & \cdots & \Phi_{s^*}(\tau_{21}) \\ \Phi_1(\tau_{31}) & \cdots & \Phi_{s^*}(\tau_{31}) \\ \Phi_1(\tau_{32}) & \cdots & \Phi_{s^*}(\tau_{32}) \\ \vdots & & \vdots \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & b_{13} & \cdots \\ b_{21} & b_{22} & b_{23} & \cdots \\ \vdots & \vdots & \vdots & \\ b_{s^*1} & b_{s^*2} & b_{s^*3} & \cdots \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots \\ 0 & \frac{1}{\gamma(\tau_{21})} & 0 & \cdots \\ 0 & 0 & \frac{1}{\gamma(\tau_{31})} & \cdots \\ 0 & 0 & \frac{1}{\gamma(\tau_{32})} & \cdots \\ \vdots & \vdots & \vdots & \end{pmatrix}$$

führen, wobei τ_{21} der einzige Baum mit zwei Knoten ist, und τ_{31} und τ_{32} die beiden Bäume mit drei Knoten bezeichnen.

Aufgabe 2b:

(a) Leiten Sie zum klassischen Runge-Kutta-Verfahren gegeben durch das Tableau

$$\begin{array}{c|cccc} 0 & 0 & & & \\ \frac{1}{2} & \frac{1}{2} & 0 & & \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

die eindeutige, stetige Runge-Kutta-Lösung mit $s^* = 4$ und $p^* = 3$ her.

Lösung:

$$b_1(\theta) = \theta - \frac{3\theta^2}{2} + \frac{2\theta^3}{3}\ , \quad b_2(\theta) = b_3(\theta) = \theta^2 - \frac{2\theta^3}{3}\ , \quad b_4(\theta) = -\frac{\theta^2}{2} + \frac{2\theta^3}{3}$$

(b) Zeigen Sie, dass die stetige Runge-Kutta-Lösung zwar global stetig, aber nicht global differenzierbar ist.

Aufgabe 2c:
Erweitern Sie Ihr Programm aus Projekt 1 um die Möglichkeit, die numerische Lösung an beliebigen, vorgegebenen Stellen ausgewertet zu bekommen. Benutzen Sie hierzu die in Aufgabe 2b berechnete stetige Runge-Kutta-Lösung.

In der Erweiterung sollte das Programm das Eingabe-Element des Templates 1 `tspan` auch als Vektor von aufsteigend sortierten Zahlen akzeptieren. Wenn der Vektor nur zwei Einträge hat, sollte die Ausgabe genau die sein, die die Schrittweitensteuerung produziert. Wenn der Vektor mehr Einträge hat, sollte die stetige numerische Lösung berechnet werden. Dazu speichern Sie nicht mehr nach jedem akzeptierten Schritt die normalen Daten, sondern überprüfen, ob ein oder mehrere Ausgabepunkte in dem aktuellen Intervall liegen. Ist das der Fall, berechnen Sie die stetige numerische

Lösung an den entsprechenden Stellen und speichern diese. Im nächsten Schritt werden die regulären numerischen Werte überschrieben.

Aufgabe 2d:

Wenden Sie die Erweiterung Ihres adaptiven Runge-Kutta-Verfahrens auf die van-der-Pole-Differentialgleichung

$$y_1'(t) = y_2(t)$$
$$y_2'(t) = \big(1 - y_1^2(t)\big)y_2(t) - y_1(t)$$

aus Aufgabe 1d auf dem Intervall $I = [0, 10]$ mit Startwert $y_0 = (2, 0)^T$ an. Geben Sie eine absolute und relative Toleranz von 10^{-1} an. Plotten Sie die Lösung, die die Schrittweitensteuerung liefert. Benutzten Sie nun das Programm mit der stetigen numerischen Lösung für Ausgaben in Abständen von 0.1 und plotten Sie diese Lösung in das selbe Bild. Wiederholen Sie das Experiment mit unterschiedlich scharfen Toleranzen und machen Sie den Fehler für die stetige numerische Lösung sichtbar.

Sprechstunde: Mitwochs von 11:00 Uhr bis 12:00 Uhr in Raum 15
Fragen an: `Julia.Schweitzer@unibas.ch`
`www.math.unibas.ch/~stohrer/vorlesungen/10hs_numdgl`