Prof. Dr. Karsten Urban
Dipl.-Math. oec. Klaus Stolle
Institut für Numerische Mathematik
Universität Ulm

WiSe 2015/2016
Blatt 6
20.11.2015

Angewandte Numerik 2

Abgabetermin: Freitag, 27.11.2015, **vor der Übung**

Für dieses Übungsblatt gibt es 10 Theorie- und 17 Matlab-Punkte, sowie 8 Theorie- und 13 Matlab-Zusatzpunkte. Punkte, die mit einem * gekennzeichnet sind, sind Zusatzpunkte.
Die 50-Prozent-Grenzen liegen aktuell (inklusive Blatt 6) bei 51 Theoriepunkten und 55,5 Matlabpunkten.

Aufgabe 25 *(Gewöhnliche Differentialgleichungen: Richtungsfeld)* (5T+3T* Punkte)

Gegeben sei die Differentialgleichung

$$y' = ay - by^3 \tag{1}$$

mit reellen Konstanten $a, b > 0$.

a) Skizzieren Sie das Richtungsfeld für $a = 4, b = 1$ in $-3 \leq t \leq 3, -3 \leq y \leq 3$.

b) Es sei y eine Lösung von (1) mit $y_0 = y(0)$. Argumentieren Sie alleine mit Hilfe des Richtungsfelds, wie sich $y(t)$ für $t \to \infty$ in Abhängigkeit von y_0 verhält.

Aufgabe 26 *(Programmieraufgabe: Verschiedene Einschrittverfahren)*
(3M+2M+2M+3M+2M+3M+2M+3M*+2M*+5T Punkte)

a) Schreiben Sie eine Matlabfunktion `yk = eulerExplizit(f, y0, tk)`, die für einen gegebenen Startwert $y^0 \in \mathbb{R}^n$ eine Lösung $y : \mathbb{R} \to \mathbb{R}^n$, $n \in \mathbb{N}$ der Anfangswertaufgabe

$$y' = f(t, y), \qquad y(t_0) = y^0$$

mit dem expliziten Euler-Verfahren

$$y^{k+1} = y^k + h_k\, f(t_k, y^k) \quad \text{mit} \quad h_k = (t_{k+1} - t_k)$$

berechnet. Der Parameter `f` ist dabei die Funktion f als *function handle*, `y0` ist der Startwert $y^0 \in \mathbb{R}^n$ und `tk` ist ein Gitter mit den diskreten Zeitpunkten t_k. Der Rückgabewert `yk` ist der Vektor der einzelnen Näherungswerte y^k.

b) Schreiben Sie ein Matlabskript `vergleichESV`, das mit Ihrer Matlabfunktion `eulerExplizit` für das Anfangswertproblem

$$y'(t) = y(t) + 2\,e^t, \quad y(0) = 2$$

auf dem Intervall $[0, 1]$ eine Näherungslösung nach dem expliziten Euler-Verfahren berechnet. Wählen Sie als Schrittweite zunächst $h = \frac{1}{20}$ und lösen Sie ebenfalls für $h = \frac{1}{40}$. Vergleichen Sie die Ergebnisse mit der exakten Lösung $y(t) = 2(t + 1)e^t$. Zeichnen Sie dazu Ihre beiden Näherungslösungen (für die zwei verschiedenen Schrittweiten) und die exakte Lösung in ein Schaubild.

c) Zeichnen Sie in ein weiteres Schaubild die beiden Gitterfehler (für die beiden Schrittweiten) ein (siehe Definition 3.2.9).

d) Schreiben Sie eine Matlabfunktion `yk = eulerImplizit(f, y0, tk, tol)`, die analog zu `eulerExplizit` die Anfangswertaufgabe mit dem impliziten Euler-Verfahren löst. Berechnen Sie die Näherung für y^{k+1} mit einer Fixpunktiteration. Die Fixpunktiteration soll abbrechen, wenn die Genauigkeit `tol` erreicht ist.

e) Bestimmen Sie mit `eulerImplizit` zwei weitere Näherungslösungen der Anfangswertaufgabe mit den Schrittweiten aus Aufgabenteil b). Zeichnen Sie diese Näherungslösungen und die exakte Lösung in ein neues Schaubild ein. Ergänzen Sie Ihr Schaubild mit den Gitterfehlern um die Gitterfehler der beiden neuen Lösungen.

f) Schreiben Sie eine Matlabfunktion `yk = trapezMethode(f, y0, tk, tol)`, die analog zu `eulerImplizit` die Anfangswertaufgabe mit dem Verfahren nach Beispiel 3.2.3 (c) aus dem Skript löst. Berechnen Sie die Näherung für y^{k+1} analog zu `eulerImplizit` mit einer Fixpunktiteration.

g) Bestimmen Sie mit `trapezMethode` wiederum zwei weitere Näherungslösungen der Anfangswertaufgabe und den Schrittweiten aus Aufgabenteil b), die Sie zusammen mit der exakten Lösung in ein neues Schaubild eintragen. Ergänzen Sie Ihr Schaubild mit den Gitterfehlern um die Gitterfehler der beiden neuen Lösungen.

h) Als letztes Verfahren sollen Sie in einer Matlabfunktion `yk = heun(f, y0, tk)` das Verfahren von Heun (zweiter Ordnung) implementieren. Dieses Verfahren berechnet die Näherungslösungen ähnlich der Trapezmethode, ist aber ein explizites Verfahren und benötigt daher keine Fixpunktiteration. In einem ersten Schritt wird der Wert y^{k+1} mit einem Schritt des expliziten Euler-Verfahrens geschätzt. In einem zweiten Schritt wird diese Schätzung in die Trapezregel eingesetzt. Die Verfahrensfunktion lautet also $F(f, t_k, h_k, y^k) = \frac{1}{2}\left(f(t_k, y^k) + f(t_k + h_k,\, y_k + h_k f(t_k, y^k))\right)$.

i) Bestimmen Sie auch mit `heun` zwei Näherungslösungen der Anfangswertaufgabe und den Schrittweiten aus Aufgabenteil b), die Sie zusammen mit der exakten Lösung in ein neues Schaubild eintragen, und ergänzen Sie Ihr Schaubild mit den Gitterfehlern um die Gitterfehler der beiden neuen Lösungen.

j) Erläutern Sie die Schaubilder. Vergleichen Sie alle vier Verfahren.

Aufgabe 27 *(Programmieraufgabe: Zeeman'sches Herzschlagmodell)* (5M*+3M*+5T* Punkte)

Zeemanns Herzschlagmodell beschreibt die Funktionsweise des Herzens. Gesuchte Größen sind die Länge der Herzmuskelfaser $l(t)$ und das elektrochemische Potential $p(t)$. Das Modell wird beschrieben durch das System

$$\frac{d}{dt}\begin{pmatrix} l(t) \\ p(t) \end{pmatrix} = \begin{pmatrix} -(l(t)^3 - \alpha l(t) + p(t)) \\ \beta l(t) \end{pmatrix},$$

wobei α die Vorspannung der Muskelfaser und β der Rückkopplungsparameter sind.

a) Berechnen Sie in einem Matlabskript `zeemann` mit Ihrer Matlabfunktion `eulerExplizit` eine Näherungslösung im Zeitintervall $[0, 100]$ mit Schrittweite $h = 0.05$. Verwenden Sie $\alpha = 3$ und $\beta = 0.1$ und die Anfangswerte $l(0) = 1$ und $p(0) = 0$. Stellen Sie Ihre Näherungslösung grafisch dar.

b) Berechnen Sie auch mit den anderen Einschrittverfahren aus Aufgabe 26 Näherungslösungen.

c) Decken sich die berechneten Ergebnisse in der graphischen Darstellung? Wie passt Ihre Beobachtung in dieser Aufgabe zu Ihren Ergebnissen aus Aufgabe 26? Welches Verfahren liefert (bzw. welche Verfahren liefern) vermutlich die genaueren Ergebnisse? Begründen Sie Ihre Vermutung.

Hinweise:
Die Programmieraufgaben sind in Matlab zu erstellen. Senden Sie alle Matlab-Files in einer E-mail mit dem Betreff **Loesung-Blatt06** an **angewandte.numerik@uni-ulm.de** (Abgabetermin jeweils wie beim Theorieteil). Drucken Sie zusätzlich allen Programmcode sowie die Ergebnisse aus und geben Sie diese vor der Übung ab. Der Source Code sollte strukturiert und, wenn nötig, dokumentiert sein.