Runde 3, Beispiel 19

LVA 118.181, Übungsrunde 3, 03.11.

Markus Nemetz, markus.nemetz@tuwien.ac.at, TU Wien, 01.11.2006

1 Angabe

Man vergleiche das Euler-Verfahren, das Verfahren von Heun und das Verfahren von Runge- Kutta für das AWP $y = 1 + \frac{y}{x}, y(1) = 1$ an der Stelle $x = 1.6$, wobei $n = 3$ gewählt werden soll. Löse das AWP auch exakt und ermittle jeweils den relativen Fehler für die einzelnen Verfahren.
Anmerkung: Nach Möglichkeit programmiere man die einzelnen Verfahren. Wenn 'mit der Hand' gerechnet werden muss, wähle $n = 1$.

2 Theoretische Grundlagen: Euler-Verfahren

Das Euler-Verfahren ist ein numerisches Vefahren zur Lösung von Anfangswertproblemen. Die Idee ist, denn Differentialquotienten $y' = \frac{\partial y}{\partial x}$ durch den Differenzenquotienten $\frac{\triangle y}{\triangle x} = \frac{y(x+h)-y(x)}{h}$ zu ersetzen (h ist die Schrittweite):

$$f(x, y(x)) = y'(x) = \frac{\partial y}{\partial x} \approx \frac{\triangle y}{\triangle x} = \frac{y(x+h) - y(x)}{h}$$

Die Auflösung nach $y(x + h)$ ergibt

$$y(x + h) \approx y(x) + hf(x, y(x))$$

Wenn wir nun den bekannten Anfangswerk $y_0 = y(x_0)$, eine kleine Schrittweite h wählen und $x = x_0$ und $y = y_0$ setzen, erhalten wir mit y_1 eine Näherung für den exakten Wert $y(x_0 + h)$:

$$y(x_1) = y(x_0 + h) \approx y_0 + fh(x_0, y_0) =: y_1$$

Daraus können wir y_2 errrechnen usw. worazs such folgende allgemeine Formel ergibt:

$$y_{n+1} = y_n + hf(x_n, y_n)$$

3 Theoretische Grundlagen: Heun-Verfahren

Das Heun-Verfahren, benannt nach Karl Heun, ist ein einfaches Verfahren zur numerischen Lösung von Anfangswertproblemen. Es ist ein Einschrittverfahren und gehört zu der Klasse der Runge-Kutta-Verfahren.
Im Gegensatz zum Explizites Euler-Verfahren erfolgt die Näherung über ein Trapez und nicht über ein Rechteck.

Zur numerischen Lösung des Anfangswert-Problems:

$$\dot{x} = f(t, x), \qquad x(t_0) = x_0$$

für eine gewöhnliche Differentialgleichung mit dem Verfahren von Heun wähle man eine Diskretisierungs-Schrittweite $h > 0$, betrachte die diskreten Zeitpunkte

$$t_k = t_0 + kh, \qquad k = 1, 2, \cdots < /math >$$

und berechne zunächst analog zum expliziten Euler-Verfahren

$$x_{k+1}^{[P]} = x_k + hf(t_k, x_k) \quad , \quad k = 1, 2, \ldots$$

und dann

$$x_{k+1} = x_k + \frac{1}{2}h(f(t_k, x_k) + f(t_{k+1}, x_{k+1}^{[P]})) \quad , \quad k = 1, 2, \ldots$$

Die x_i sind die Näherungswerte der tatsächlichen Lösungsfunktion $x(t)$ zu den Zeitpunkten t_i.
h bezeichnet man als Schrittweite; Verkleinert man die Schrittweite so wird der Verfahrensfehler kleiner (sprich die x_i liegen näher am tatsächlichen Funktionswert $x(t_i)$). Der globale Fehler des Verfahren von Heun geht mit h^2 gegen Null man spricht auch von Konvergenzordnung 2.

4 Theoretische Grundlagen: Runge-Kutta-Verfahren

klassische Runge-Kutta-Verfahren ist ein explizites 4-stufiges Einschrittverfahren der numerischen Mathematik zur näherungsweisen Lösung von Anfangswertproblemen. Der Name stammt von den deutschen Mathematikern Carl Runge und Martin Wilhelm Kutta-Das klassische Runge-Kutta-Verfahren verwendet - wie die weitaus meisten numerischen Lösungsverfahren für Differentialgleichungen - den Ansatz, Ableitungen (Differentialquotient) durch (endliche) Differenzenquotienten zu ersetzen. Die dabei bei nichtlinearen Funktionen notwendigerweise auftretenden Fehler (es werden sämtliche höheren Glieder der Taylor-Entwicklung vernachlässigt) können durch geeignete Kombinationen verschiedener Differenzquotienten teilweise kompensiert werden. Das Runge-Kutta-Verfahren ist nun eine solche Kombination, die Diskretisierungsfehler bis zur dritten Ableitung kompensiert.
Es sei

$$y' = f(x, y)$$

eine (gewöhnliche) Differentialgleichung 1.Ordnung mit den Anfangsbedingungen

$$y(x_0) = y_0$$

und sei weiter h die gewünschte Schrittweite.
Dann wird der (angenäherte) Wert von $y(x_0+h) = y(x_1)$ nach Runge wie folgt errechnet:

x	y	y'
x_0	y_0	y'_0
$x_0 + \frac{h}{2}$	y_A	y'_A
	y_B	y'_B
$x_0 + h$	y_C	y'_C
$x_0 + h = x_1$	y_1	y'_1

Hinweis: In der Literatur wird häufig an Stelle der y'_X direkt die Schrittdifferenz $k_X = h.y'_X$ angegeben. Das ist insbesondere hinsichtlich der in den folgenden Abschnitten angegebenen Anwendung auf Differentialgleichungen höherer Ordnung unpraktisch und wird darum (weil es die schematische Rechenarbeit unterbricht und auch sonst keinen Gewinn bringt) hier nicht verwendet.

$$\begin{array}{rcc} y'_0 & = & f(x_0, y_0) \\ y_A & = & y_0 + \frac{h}{2} \cdot y'_0 \\ y'_A & = & f(x_0 + \frac{h}{2}, y_A) \\ y_B & = & y_0 + \frac{h}{2} \cdot y'_A \\ y'_B & = & f(x_0 + \frac{h}{2}, y_B) \\ y_C & = & y_0 + h \cdot y'_B \\ y'_C & = & f(x_0 + h, y_C) \\ y_1 & = & y_0 + h \cdot \frac{y'_0 + 2(y'_A + y'_B) + y'_C}{6} \end{array}$$

Mit den neuen Werten x_1 und y_1 kann dann der nächste Rungeschritt durchgeführt werden. Die erzielte Genauigkeit liegt - bei genügend glattem $f(x, y)$ - in der Größenordnung von h^4.

5 Lösung des Beispiels

$$y = 1 + \frac{y}{x} \qquad \Rightarrow \qquad y - \frac{y}{x} = 1$$

Eine Möglichkeit zur Lösung besteht in der **Verwendung einer Formel**: Lineare inhomogene Differentialgleichungen 1. Ordnung in der Form $y' + p(x)y = r$ (r ist Störfunktion, singulär oder nur von x abhängig) können mit folgender Formel aufgelöst werden:

$$h = \int p(x)\, dx$$

$$y(x) = e^{-h}(\int e^h r\, dx + c)$$

(Quelle: Kreyszig, Advanced Engineering Mathematics, 9.Aufl., S.26ff.)

$$r = 1, \qquad p = -\frac{1}{x}, \qquad h \int p \; dx = -\ln x$$

$$y(x) = \underbrace{e^{\ln x}}_{x} \cdot (\int \underbrace{e^{-\ln x}}_{\frac{1}{x}} \cdot 1 + c)$$

$$y(x) = x \cdot (\int \frac{1}{x} \cdot 1 + c)$$

$$y(x) = x \cdot (\ln x + c)$$

$$y(x) = x \cdot \ln x + c$$

Spezielle Lösung für $y(1) = 1$:

$$y(x) = x \cdot \ln x + x$$

Ausprogrammiert in MATLAB:

Listing 1: Definition der Funktion

```
%  Funktion phi=f(t,y) ist die rechte Seite von DGL y'(t)=f(t,y).
%  Laut der MATLAB-Syntax muss dann die Datei den Name "f.m" haben.

  function phi=f(t,y)

    phi=1 + t./y;
```

Listing 2: numdgl.m

```
%  Numerische Loesung der DGL y'(t)=f(t,y) mit dem
%    *  expliziten Euler-Verfahren
%    *  verbesserten Euler-Verfahren (Euler-Heun)
%    *  Runge-Kutta-Verfahren 4. Ordnung.
%
%  Die Funktion f(t,y) ist in der datei "f.m" angegeben.

   clear all;

   h=0.1;     %  Schrittweite
   N=10;     %  Anzahl der Schritte
   t=1.6;         %  Anfangszeit
   T=[t:h:t+h*N];     %  Array fuer Graphen
   y0=1;     %  Anfangswert
   YEX=t.*log(t) +t;     %  Exakte Loesung


   yee=y0;     %  yee ist die numerische Loesung mit dem expliziten Euler-Verfahren
   YEE(1)=y0;     %  YEE ist das Array fuer Graphen

   yeh=y0;      %  yeh ist die numerische Loesung mit dem Euler-Heun-Verfahren
   YEH(1)=y0;     %  YEH ist das Array fuer Graphen

   yrk=y0;      %  yrk ist die numerische Loesung mit dem Runge-Kutta-Verfahren
   YRK(1)=y0;     %  YRK ist das Array fuer Graphen


```

```
28    for n=1:N
29
30
31 %  Euler-Explizit
32
33      yee=yee+h*f(t,yee);    %  Ein Schritt des expliziten Euler-Verfahrens
34      YEE(n+1)=yee;
35
36 %  Euler-Heun
37
38      k1=f(t,yeh);
39      k2=f(t+h/2, yeh+h/2*k1);
40      yeh=yeh+h*k2;        %  Ein Schritt des Euler-Heun-Verfahrens
41
42      YEH(n+1)=yeh;
43
44
45 %  Runge-Kutta
46
47      k1=f(t,yrk);
48      k2=f(t+h/2, yrk+h/2*k1);
49      k3=f(t+h/2, yrk+h/2*k2);
50      k4=f(t+h, yrk+h*k3);
51      yrk=yrk+h/6*( k1+2*(k2+k3)+k4 );        %  Ein Schritt des Runge-Kutta-Verfahrens
52      YRK(n+1)=yrk;
53
54
55
56 %  Naechster Zeitschritt
57
58      t=t+h;
59
60    end
61
62
63 %  Graphen
64
65    plot(T, YEX, 'k');
66    hold on
67    plot(T,YEE,'r');
68    plot(T,YEH,'g');
69    plot(T,YRK,'b');
70    hold off
71
72    title('Numerische Loesung von DGL');
73    xlabel('x');
74    ylabel('Loesung y');
75    legend('Exakte Loesung','Euler-Explizit','Euler-Heun','Runge-Kutta');
```