Euler-Verfahren, Verlet-Verfahren

Jörn Loviscach

Versionsstand: 16. Mai 2009, 23:08

1 Numerische Lösung von Differentialgleichungen

Die DGLn der meisten interessanten Modelle können *nicht* mit Bleistift und Papier gelöst werden (etwa mittels Variation der Konstanten oder mittels Trennung der Variablen). Die Lösungsfunktionen von DGLn lassen sich typischerweise nicht einmal mit Standardfunktionen wie sin und exp hinschreiben. Also versucht man, die Lösungsfunktionen *numerisch* zu bestimmen: Wertetabellen aufzustellen. Das geht immer nur mehr oder minder genau!

Demo für eine Physik-Simulation mit Maxon Cinema 4D.

Die DGL wird in eine Software gefüttert (numerischer DGL-Löser [numerical integrator]), die dann so eine Wertetabelle bestimmt, vgl. die Funktion `lsode` in Octave. Typischerweise erwarten DGL-Löser eine DGL in dem Format

$$y'(x) \stackrel{!}{=} f(y(x),x) \quad \text{mit} \quad y(x_0) \stackrel{!}{=} y_0,$$

wobei der Startpunkt $(x_0|y_0)$ vorgegeben ist. Die Funktion f wird „programmiert“, so dass der DGL-Löser sie für alle $(x|y)$ auswerten kann, für die er das will. Achtung: Die DGL wird auch gerne als $\dot{x} \stackrel{!}{=} f(x,t)$ geschrieben, also mit t statt x und x statt y. Also Vorsicht mit der Rolle von x.

Beispiel: Was ist f für die folgende DGL?

$$x^2 y' + \sin(y) - x^5 \stackrel{!}{=} 0$$

[1]

In der nächsten Vorlesung geht es darum, wie man DGL höherer Ordnung auch so behandeln kann.

2 Explizites Euler-Verfahren

Gegeben ist also eine DGL mit Anfangsbedingung:

$$y' \stackrel{!}{=} f(y,x) \quad \text{mit} \quad y(x_0) \stackrel{!}{=} y_0.$$

Gesucht ist die Funktion $y(x)$ für $x \geq x_0$.

Der Wert von $y(x_0)$ ist vorgegeben. Wenn man in x nun ein kleines Stückchen h weitergeht, also zu $x = x_0 + h$, kennt man für sehr kleines h die Änderung von y in guter Näherung und kann damit sagen:

$$y(x_0 + h) \approx y_1 := \boxed{2} .$$

Nun kann man schätzen, was y sein soll, wenn man noch einen Schritt h weiter geht:

$$y(x_0 + 2h) \approx y_2 := \boxed{3} .$$

Und so weiter. Dies ist das „explizite Euler-Verfahren“.

Demo mit OpenOffice.org mit $(x_0|y_0) = (2|3)$ und der Bestimmung von $y(10)$ mit Schrittweiten von 1 bis hinunter zu 0,005. Wenn man die Schrittweite h halbiert, verdoppelt sich die Zahl der Schritte, um einen bestimmten x-Wert von x_0 aus zu erreichen. Andererseits halbiert sich dabei ungefähr der Fehler der Näherung. Das stößt allerdings an eine Grenze, wenn die Schrittweite h zu klein wird: Dann addiert man in dem Iterationsschritt Zahlen mit zu verschiedenen Größenordnungen, so dass die Rundungsfehler überhand nehmen. Beispiel:

4

3 Andere Löser für DGLn erster Ordnung

Um das $y(x)$ zu einem vorgegeben x sehr genau zu bestimmen und auch noch schnell zu rechnen, benötigt man deshalb Verfahren, die mit recht groben Schrittweiten h auskommen. Das übliche Verfahren ist das vierter Ordnung von Runge-Kutta. Wenn man hier die Schrittweite halbiert, sinkt der Fehler etwa auf ein Sechzehntel. Allerdings muss man die Funktion f dazu pro Schritt viermal auswerten.

Neben der Genauigkeit von $y(x)$ zu einem vorgegeben x kann man auch das Verhalten für $x \to \infty$ untersuchen: Die genäherte Lösung soll nicht explodieren. Hierzu kann man zum Beispiel *implizite* Löser einsetzen: Beim *impliziten* Euler-Verfahren setzt man schon den neuen y-Wert y_1 rechts ein:

$$y(x_0 + h) \approx y_1 := \boxed{5} .$$

Diesen neuen Wert y_1 hat man aber noch gar nicht und muss deshalb diese Gleichung erst nach y_1 auflösen.

Implizite DGL-Löser sind auch hilfreich beim Umgang mit steifen [stiff] DGL-Systemen. So heißen DGL-Systeme, in denen sehr verschieden schnelle Abklingverhalten gemischt werden. (Beispiele?) Die Anteile mit dem schnellen Abklingverhalten erzwingen gegebenenfalls eine entsprechend kleine Schrittweite.

4 Verlet-Verfahren

Gerade in der Physik ist man an DGLn zweiter Ordnung interessiert. Im Prinzip lassen sich diese zwar in DGLn erster Ordnung umwandeln, wie schon beim Federpendel vorgeführt. Aber alternativ kann man sich auch die Struktur der Differentialgleichungen der Mechanik zu Nutze machen. Ein Massepunkt der Masse m befinde sich in einem Kraftfeld $F(\mathbf{x}, t)$. Das führt auf folgende Differentialgleichungen:

$$\dot{\mathbf{x}} \stackrel{!}{=} \boxed{6}$$

$$\dot{\mathbf{p}} \stackrel{!}{=} \boxed{7}$$

zu den Anfangsbedingungen $\mathbf{x}(t_0) \stackrel{!}{=} \mathbf{x}_0$ und $\mathbf{p}(t_0) \stackrel{!}{=} \mathbf{p}_0$.

Mit dem expliziten Euler-Verfahren sähe der erste Schritt von t nach $t+h$ so aus:

8

Demo mit OpenOffice.org für ein ungedämpftes Federpendel.

Witzigerweise stellt sich heraus, dass der DGL-Löser wesentlich besser funktioniert, wenn man das *neue* $\mathbf{x}$ in die Gleichung für $\mathbf{p}$ einsetzt – oder umgekehrt: das *symplektische* Euler-Verfahren:

9

Demo mit OpenOffice.org.

Noch besser funktioniert das Verlet-Verfahren. Hier ist dessen Leapfrog [Bocksprung] benannte Unterart: Man setzt einfach an, dass sich $\mathbf{x}$ von einem Schritt zum nächsten so weiter entwickelt, wie es das im Schritt davor getan hat – aber nun korrigiert durch die Wirkung der Beschleunigung über das Zeitintervall $2h$:

10

Nur im allerersten Schritt hat man keine Vorgängerposition. Dort nimmt man die Anfangsgeschwindigkeit:

11

Demo mit OpenOffice.org.

Sowohl das symplektische Euler-Verfahren wie das Verlet-Verfahren erhalten zwar nicht genau die Energie, aber dafür eine Größe, die gegenüber der Energie nur etwas gestört ist. Das stellt sicher, dass die berechneten Bahnen nicht aus dem Ruder laufen. Diese Garantie hat man typischerweise nicht, wenn man eines der allgemeinen Verfahren (Euler, Runge-Kutta usw.) auf physikalische Probleme anwendet.