"Erste Schritte mit DSTOOL"

Jörg Härterich
Freie Universität Berlin

April 25, 2002

Diese Kurzanleitung soll dazu dienen, sich in ca. 30 Minuten mit den wichtigsten Möglichkeiten von DSTOOL bekannt zu machen.

1 Installation

In dieser Anleitung wird die Benutzung der Multi-User-Version erklärt, die im Verzeichnis
`/home/diophant/dynamik/public/dstool2`
installiert ist.
Um auf diese Version zuzugreifen, müssen zwei Environment-Variablen neu definiert werden. Bei Benutzung von cshell (das ist am Fachbereich voreingestellt) fügt man dazu in den File `.cshrc` die beiden Zeilen

```
setenv DSTOOL /home/diophant/dynamik/public/dstool2
setenv DSTOOL_COLOR_DIR /home/diophant/dynamik/public/dstool2/colormaps
```

ein und erweitert die Liste der Pfadnamen, die nach `PATH` angegeben sind um `$DSTOOL`.

Achtung: Vor dem Editieren Sicherungskopie von `.cshrc` anfertigen.
Anschließend mit

```
exec csh
```

die neuen Variablen einlesen. (Ab dem nächsten Mal geschieht dies dann automatisch beim login.)
Nun kann man DSTOOL durch den Befehl `dstool` starten und es erscheint oben links ein Fenster, das im Folgenden "Hauptfenster" heißt.

2 Voreingestellte Dynamische Systeme

DSTOOL kennt schon etwa zehn Dynamische Systeme, die man mit dem Menü **Models** auswählen kann. Wähle unter "Standard Vector Fields" das Modell "Planar Cubic Vector Field" aus. Dieses Modell simuliert die Differentialgleichung

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= -(x^3 + rx^2 + nx + m) + y(b - x^2) \end{aligned}$$

mit voreingestellten Parametern $r = n = m = b = \frac{1}{10}$. Klicke mit der linken Maustaste nun auf den Knopf **View** im Hauptfenster (alternativ kann man auch aus dem Menü "View" den Punkt "2D View" auswählen). Es erscheint ein neues Fenster, in dem die Ergebnisse der Simulation dargestellt werden.

Nun können wir die ersten Trajektorien berechnen. Fahre mit der Maus in das Fenster und drücke

nacheinander die linke und die mittlere Maustaste. Durch die linke Taste wird der Anfangspunkt festgelegt, beim Drücken der mittleren Taste startet die Simulation. Experimentiere mit anderen Anfangsbedingungen, bis Du eine Vorstellung davon hast, wie das Phasenportrait aussieht.
Klicke mit der linken Maustaste im Hauptfenster auf die beiden Knöpfe **Orbits** und **Settings**. Es erscheinen zwei weitere Fenster. Im Fenster "Selected Point" kann man von Hand die Anfangsbedingung für die Simulation und den Wert der Parameter vorgeben. Ändere den Parameterwert von $b = 0.1$ auf $b = 0.5$. Mit den Knöpfen **Forward** und **Backward** im "Orbit"-Fenster (oder per Mausklick) startet man dann die Simulation in positiver, bzw. negativer Zeitrichtung.

Im "Orbit"-Fenster kann man noch weitere Einstellungen vornehmen, beispielsweise mit "Step Size" die Größe der Zeitschritte vorgeben oder unter "Stop" die Anzahl der Schritte. Lösche mit **Clear All** die bisher gezeichneten Trajektorien. Setze dann zunächst "Stop=2500", "Stepsize=0.05" und starte die Simulation mit Anfangsbedingungen $x = y = 3$. Nun wollen wir das Bild dieser Trajektorie ausdrucken: Wähle dazu im Menü "Options" den Punkt "Print..." aus.

3 Kontinuierliche Systeme: Volterra-Lotka-Modelle

Meist wollen wir nicht die in `DSTOOL` vorhandenen Gleichungen benutzen, sondern selbst definierte Dynamische Systeme simulieren.
Starte dazu `DSTOOL` und wähle im Menü **Model** zunächst "Parser", dann "Parsed Dynamical System...". Es erscheint ein kleines Fenster mit einigen Zeilen Text:

```
# The Lorenz system
x' = sigma ( y - x )
y' = rho x - y - x z
z' = -beta z + x y

INITIAL sigma 10.0 rho 28.0 beta 2.66666666
RANGE x -30 30 y -30 30 z -20 50
```

Dieser Text enthält Informationen über die Differentialgleichung, Anfangsbedingungen ("INITIAL"), sowie den darzustellenden Bereich ("RANGE"). Hier ist das die Lorenz-Gleichung

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy \end{aligned}$$

mit den Parameterwerten $\sigma = 10$, $\rho = 28$ und $\beta = \frac{8}{3}$.
Ersetze den Text durch:

```
# A Lotka-Volterra Model
x' = x - 2*x*y
y' = -3*y + 0.5*x*y
INITIAL x 1.2 y 0.4
RANGE x 0 3 y 0 3
```

Drücke dann den Knopf **Build**, um das neue Modell einzulesen.
Achtung: Nicht den Knopf **Make C Code** drücken, sonst beendet ein ungeklärter Fehler sofort das Programm !!!

Öffne jetzt das "View"-Fenster und berechne per Mausklick einige Trajektorien. Vergrößere dann den

Bereich auf $0 \leq x \leq 10$, $0 \leq y \leq 15$. Anschließend **Refresh** drücken, erst dann ist die neue Größe eingestellt ! Berechne eine weitere Trajektorie. Sind die Trajektorien wirklich alle periodisch ?
Wie groß kann man den Zeitschritt "Step Size" wählen, bevor man mit dem bloßen Auge eine Abweichung von der periodischen Bahn schon bei einem Umlauf sehen kann ?
Trage dann im "View"-Fenster horizontal die Zeit t ("`time`") und vertikal $x(t)$ auf.

4 Die logistische Abbildung

In diesem Abschnitt wollen wir ein diskretes dynamisches System simulieren, die logistische Abbildung

$$x_{n+1} = \lambda x_n (1 - x_n)$$

mit $x \in [0, 1]$ und einem Parameter $\lambda \in [0, 4]$.
Gib dazu im Parser-Fenster den Text

```
# Logistic map
x' = lambda*x*(1-x)

INITIAL x 0.43 lambda 3.4
RANGE x 0 1 lambda 0 4
```

ein, wähle außerdem **Mapping** statt **Vector Field** und drücke **Build** zur Bestätigung.
Öffne dann mit **View** ein Fenster und stelle die Achsen so ein, dass λ horizontal und x vertikal aufgetragen werden. Starte dann per Maus eine Simulation. Da bei der Iteration λ konstant bleibt, sollten alle Punkte übereinander liegen. Experimentiere mit verschiedenen Startwerten (λ, x).
Wähle nun im Fenster "Selected Point" den Startwert $x = 0.01$, $\lambda = 3.2$, im Orbit-Fenster "Stop=100" und zeichne die Iterierten x_n über n ("`iter`").
Achtung: der voreingestellte Bereich für `iter` ist $0 \leq n \leq 10000$. Um etwas zu sehen, muss man den Bereich auf $0 \leq n \leq 100$ verkleinern (**Refresh** nicht vergessen).

Wenn man nur am Langzeitverhalten interessiert ist, möchte man gelegentlich die ersten Iterationen unterdrücken. Wähle dazu im Orbit-Fenster "Start=100" und "Stop=200", dann werden nur $x_{100}, x_{101}, \ldots x_{200}$ ausgedruckt.
Ändere nun λ auf $\lambda = 3.5$, $\lambda = 3.55$ bzw. $\lambda = 3.8$ und betrachte wieder die Iterierten $x_{100}, x_{101}, \ldots x_{200}$.
Was ist zu erkennen ?

Um die Abhängigkeit des Verhaltens von λ systematischer zu untersuchen, benutzen wir die Möglichkeit, unter `DSTOOL` mehrere Orbits zu verschiedenen Anfangswerten zu berechnen.
Stelle die Variablen im "View"-Fenster wieder so ein, dass horizontal λ und vertikal x aufgetragen wird. Wähle dann im Menü "Panels des Hauptfensters "Multiple..." aus. Damit können auf einmal die Trajektorien zu mehreren Anfangswerten berechnet werden. In dem nun erscheinenden Fenster, gib in der Zeile, in der links `lambda` steht bei `Size` 1.4 und bei `Number of Points` 40 ein. Setze außerdem im "Selected"-Fenster den Wert von `lambda` auf 2.6.
Dies bedeutet, dass DSTOOL automatisch 40 Anfangswerte gleichverteilt mit $2.6 - 1.4 \leq \lambda \leq 2.6 + 1.4$ wählt, also im Intervall $[1.2, 4]$.
Diese Anfangswerte werden durch Drücken von **Load Points** geladen und mit **Forwards** beginnt DSTOOL zu rechnen. Es sollten dann für 40 Werte von λ jeweils einige Punkte gedruckt werden.
Wiederhole das Ganze mit 400 Werten von λ im Intervall $[2, 4]$, um eine höhere Auflösung zu erhalten.

Beende das Programm,durch Schließen des Hauptfenster (nicht sehr elegant, aber so geht's...).

Hinweise zum DSTOOL-Parser:

1. Mit den Knöpfen **Vector Field** bzw. **Mapping** gibt man an, ob es sich um ein kontinuierliches oder ein diskretes dynamisches System handelt.
   Achtung: Auch bei diskreten Systemen steht im Parser `x'=...` etc.

2. Der Parser versteht nur Gleichungen 1. Ordnung, Gleichungen höherer Ordnung müssen als System 1. Ordnung umgeschrieben werden,

3. ebenso müssen nicht-autonome Gleichungen in ein autonomes System umgeschrieben werden.

4. Der Parser versteht **nur** die folgenden Befehle:

   ```
   Konstanten: PI, pi, E
   Operatoren: +, -, *, /, % (mod), ^
   Mathematische Funktionen: sin, cos, tan, asin, acos, atan, sinh, cosh,
           tanh, log, ln, exp, abs, sqrt
   ```

5. Mit `INITIAL` gibt man die voreingestellten Anfangsbedingungen, mit `RANGE` den voreingestellten Darstellungsbereich an. Beides kann man natürlich später ändern. Mit `PERIODIC` kann man einzelne Variablen periodisch machen, um Systeme auf der Kreislinie oder auf dem Torus zu simulieren. `PERIODIC x 0 1 y 0 1` macht die Variablen x und y beide 1-periodisch.

   Hilfsfunktionen kann man *nach* der Definition des dynamischen Systems eingeben, wie die Gleichungen selbst, jedoch ohne Strich, also z.B. `Ham = 0.5*(x*x + y*y)`.

6. Das neue dynamische System wird durch den Button **Build** geladen. Der Button **Make C Code** führt zum Absturz des Programms...