Die TEXnische Komödie

DANTE
Deutschsprachige
Anwendervereinigung TEX e.V.

Laura E. Jackson , Herbert Voß: *Die Plotfunktionen von* ***pst-plot***, Die TEXnische Komödie 2/2002, S. 27-33.

Die TEXnische Komödie ist die Mitgliedszeitschrift von DANTE, Deutschsprachige Anwendervereinigung TEX e.V. Einzelne Hefte können von Mitgliedern bei der Geschäftsstelle von DANTE, Deutschsprachige Anwendervereinigung TEX e.V. erworben werden. Mitglieder erhalten Die TEXnische Komödie im Rahmen ihrer Mitgliedschaft.

Die Plotfunktionen von `pst-plot`

Laura E. Jackson , Herbert Voß

Im letzten Heft wurden die mathematischen Funktionen von PostScript im Zusammenhang mit dem LATEX-Paket `pst-plot` zum Zeichnen von Funktionen beschrieben und durch Beispiele erläutert. In diesem Teil werden die bislang nur erwähnten Plotfunktionen für externe Daten behandelt.

Einführung

Die graphische Darstellung externer Datensätze gehört zu den Standardproblemen von technisch-wissenschaftlichen Veröffentlichungen. Sehr häufig werden diese mit `gnuplot` eingelesen, dargestellt, mit Koordinatenachsen und weiteren Hinweisen versehen und dann abschließend nach LATEX exportiert. Im folgenden werden Wege aufgezeigt, die den Umweg über `gnuplot` oder entsprechende Programme überflüssig machen.

Die Geschichte und die Bedeutung von PostScript wurden hinreichend im letzten Heft behandelt. [3] Ebenso wie das benötigte Paket `pst-plot` (`CTAN:/graphics/pstricks/latex/pst-plot.sty`), welches Teil des Pakets `pstricks` ist und mit der Anweisung `\usepackage{pst-plot}` einzubinden

ist. Daher braucht an dieser Stelle nicht weiter auf diese spezifischen Dinge eingegangen zu werden.

Das Paket stellt die folgenden drei Plotfunktionen für die Darstellung externer Daten mit folgender Syntax zur Verfügung:

```
\fileplot*[<Parameter>]{<Dateiname>}
\dataplot*[<Parameter>]{<Befehle>}
\listplot*[<Parameter>]{<Liste>}
```

Die Sternversionen stehen hierbei wie für alle Objekte von `PSTricks` jeweils für die inverse Darstellung der Daten, d.h. für eine Schwarz-Weiss-Grafik erhält man mit der Sternversion das Negativ. Im folgenden wird nur die normale Darstellung berücksichtigt, da dies keine Einschränkung der Allgemeinheit bedeutet.

Weitere Informationen kann man der Dokumentation zu `pstricks` entnehmen [5] oder der umfangreichen Beschreibung in [2, 1, 4], die jedoch alle nicht die wesentlichen Unterschiede zwischen den drei Plotfunktionen herausarbeiten. Für alle Beispiele wird jeweils die komplette `pspicture`-Umgebung angegeben, so dass eine direkte Übernahme der Beispiele möglich ist. Eine Dokumentation des `multido`-Befehls findet man unter `CTAN:/macros/latex209/contrib/multido/multido.doc`, alle anderen hier nicht weiter erwähnten in der Dokumenation zu `pstricks`. [5]

Insbesondere der Plotstil ist als Parmeter von Bedeutung und kann folgende Werte annehmen:

Stilparameter	Bedeutung
`plotstyle=dots`	Wertepaare als einzelne Punkte setzen
`=line`	Wertepaare durch eine Linie verbinden
`=polygon`	Wie `line`, nur mit geschlossenem Linienzug
`=curve`	Interpolation zwischen den Wertepaaren, wobei die Kurve über den Anfangs-/Endpunkt hinausgehen kann
`=ecurve`	Wie `curve`, nur beginnt/endet die Kurve bei dem Anfangs-/Endwertepaar.
`=ccurve`	Wie `curve` nur mit geschlossenem Kurvenzug

Bei einer fehlenden Angabe wird grundsätzlich die `line`-Option gewählt.

Weiterhin sind im Zusammenhang noch folgende Befehle interessant, die ebenfalls durch das Paket `pst-plot` definiert sind:

```
\readdata{<Objectname>}{<Dateiname>}
```

```
\savedata{<Objectname>}{<Dateiname>}
```

Beispiele für \fileplot

fileplot ist immer dann angebracht, wenn in einer Datei gespeicherte Zahlenpaare $(x|y)$ geplottet werden sollen. Diese sind als reine Zahlenwerte paarweise in einer oder meheren Zeilen anzuordnen und dürfen nur auf vier verschiedene Arten getrennt sein (Leerschritt, Komma oder runde bzw. geschweifte Klammern):

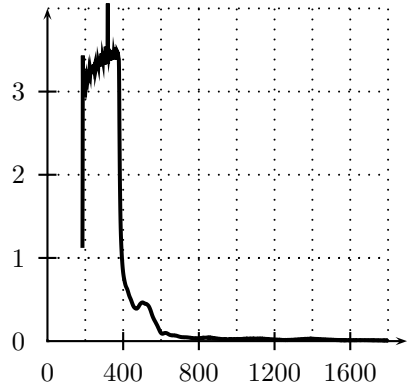
```
x y
x,y
(x,y)
{x,y}
```

Der als Trenner häufig benutzte Tabulator (\t bzw. \009) ist demnach nicht zulässig, kann aber leicht durch Texteditoren oder für Unix(e) mit

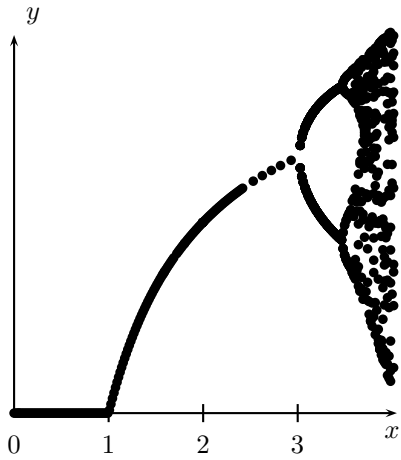
```
tr '\t' ' ' < inFile > outFile
```

ersetzt werden. Diese Datendateien dürfen bis auf das TEX-übliche Kommentarzeichen „%“ keine anderen Zeichen ausser den Zahlenwerten selbst enthalten.

Das erste Beispiel zeigt ein UV/VIS-Absorptionsspektrum ($A = \lg \frac{I_0}{I}$ als Funktion der Wellenlänge), während das zweite eine Populationsentwicklung in Abhängigkeit des Brutfaktors darstellt (Feigenbaum-Diagramm). Aus dem jeweils angegebenen Quellcode ergibt sich die Art des Plotstils.




```
\psset{xunit=0.003cm,yunit=1.5cm}
\begin{center}
\begin{pspicture}(-0.25,-.25)(1950,4)
  \fileplot[plotstyle=line]{%
    fileplot.data}
  \psaxes[dx=400,Dx=400]{->}(1900,4)
  \multido{\n=200+200}{9}{%
    \psline[linestyle=dotted](\n,0)(\n,4)
  }
  \multido{\n=+1}{5}{%
    \psline[linestyle=dotted]%
       (0,\n)(1800,\n)
  }
\end{pspicture}
\end{center}
```




```
\psset{xunit=1.5cm,yunit=6cm}
\begin{pspicture}(-0.25,-.25)(4.25,1)
  \fileplot[plotstyle=dots]{%
     feigenbaum.data}
  \psaxes{->}(0,0)(4.05,1)
\end{pspicture}
```

`fileplot` hat den Vorteil der einfachen Anwendung, aber den wesentlichen Nachteil extrem speicherintensiv zu sein, denn TEX muss erst alle Zahlenpaare vor einer weiteren Verarbeitung laden. Man kann davon ausgehen, dass man bei mehr als 1000 Punkten Schwierigkeiten mit der standardmäßig eingestellten TEX-Speicherkapazität bekommen kann. Weiterhin erhöht sich durch das Laden der Zahlenpaare die Zeit eines Kompilationsvorgangs ganz erheblich.

Eine andere Möglichkeit Speicherproblemen aus dem Weg zu gehen ist die Anwendung der `\PSTtoEPS`-Funktion auf die hier jedoch nicht weiter eingegangen wird (siehe dazu [5]).

Beispiel für `\dataplot`

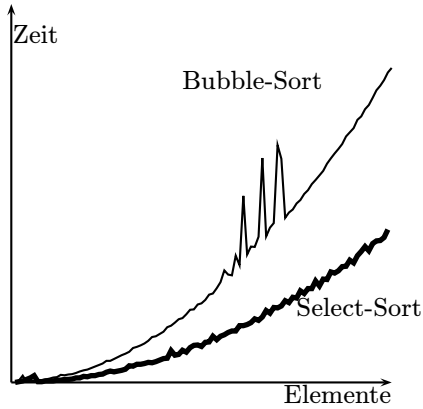
Ebenso wie `\fileplot` benötigt `\dataplot` einen externen Datensatz, der durch eine andere Anwendung erstellt wurde. Im Gegensatz dazu kann dieser externe Datensatz jedoch nicht mit `\dataplot` selbst geladen werden. Dies erfolgt mit dem oben angegebenen Befehl`\readdata`, z.B.

```
\readdata{\feigenbaum}{feigenbaum.data}
```

Die Zahl der eingelesenen Dateien ist nur duch den Speicher bestimmt, d.h. dass mit `\dataplot` einfache Overlays möglich sind. Die Ausgabe dieser Daten erfolgt mit `\dataplot{<Objektname>}` z.B. `\dataplot{\feigenbaum}`.

Das angegebene Beispiel zeigt zwei getrennte Datendateien, die in einem Koordinatensystem dargestellt werden. Es handelt sich um die Sortierzeiten der

Verfahren „Bubble-Sort“ und „Select-Sort“ in Abhängigkeit der zu sortierenden Elemente, wobei das Verfahren der Parameter ist. Die Anordnung der Elemente war zu Beginn zufällig verteilt.




```
\psset{xunit=0.0005cm,yunit=0.005cm}
\begin{center}
\begin{pspicture}(0,0)(10000,1000)
  \readdata{\bubble}{bubble.data}
  \readdata{\select}{select.data}
  \dataplot[plotstyle=line]{\bubble}
  \dataplot[plotstyle=line]{\select}
  \psline{->}(0,0)(10000,0)
  \psline{->}(0,0)(0,1000)
  \rput[l](20,995){Zeit}
  \rput[r](9990,-20){Elemente}
  \rput[l](4500,800){Bubble-Sort}
  \rput[l](7500,200){Select-Sort}
\end{pspicture}
\end{center}
```

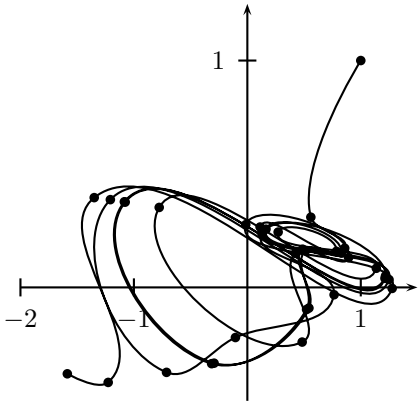
Grundsätzlich bleibt festzustellen, dass für den reiner Anwender zwischen `dataplot` und `fileplot` formal kein wesentlicher Unterschied besteht. Bei größeren Datenmengen bringt `dataplot` den Vorteil der schnelleren Verarbeitung und Darstellung, wobei es dafür aber noch speicherplatzintensiver als `fileplot` ist.

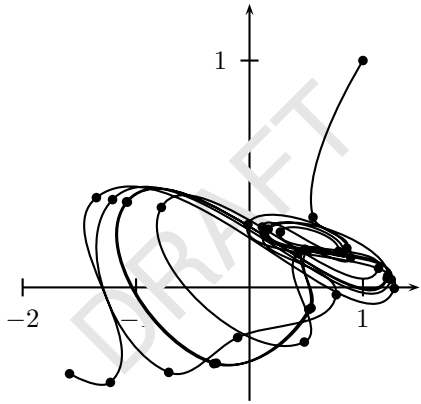
Weiterhin benutzt `\dataplot` intern die im nächsten Abschnitt beschriebene `\listplot` Funktion, wenn Parameter angegeben werden. Daraus folgt, dass `\dataplot` letztlich nur Sinn macht, wenn Polygonzüge gezeichnet werden sollen. In diesem Fall zeichnet sich diuese Funktion durch eine größere Plotgeschwindigkeit aus.

Beispiel für `\listplot`

Im Gegensatz zu den vorhergehenden Plotfunktionen wird das Argument von `\listplot` zuerst von TEX expandiert, wenn es sich um ein TEX-Befehl handelt, andernfalls wird es unverändert nach PostScript durchgereicht. Daraus folgt, dass man komplette PostScript-Programme im Argument von `\listplot` ablegen kann. Diese Möglichkeit der Kombination von TEX und PostScript bleibt jedoch in der gesamten Literatur zu `PSTricks` faktisch unberücksichtigt. Dies liegt wohl auch daran, dass `PSTricks` sehr PostScript-nah programmiert wurde, so dass sich letztlich für jede Anwendung entsprechende `PSTricks`-Befehle finden lassen.

Das angegebene Beispiel zeigt in der Originaldarstellung die Entwicklung eines Henon-Attraktors. Die rechte Grafik enthält zusätzlich zum normalen Datensatz einen durch zusätzlichen PostScript-Code erzwungenen „Draft“-Hinweis. Hierbei ist zum Verständnis allerdings eine Kenntnis der PostScript-Befehle unabdingbar, wenn dies fehlerfrei erfolgen soll. Der angegebene Sourcecode enthält aus Platzgründen nicht die eigentlichen Daten, die lediglich aus einer Folge von Zahlenpaaren bestehen, die durch „space“ voneinander zu trennen sind.






```
\readdata{\henon}{henon.dat}
\psset{xunit=1.5cm, yunit=3cm}
\begin{pspicture}(-3,-0.75)(2.25,1.5)
  \psaxes{->}(0,0)(-2,-0.5)(1.5,1.25)
  \listplot[showpoints=true,%
     plotstyle=curve]{\henon}
\end{pspicture}
```

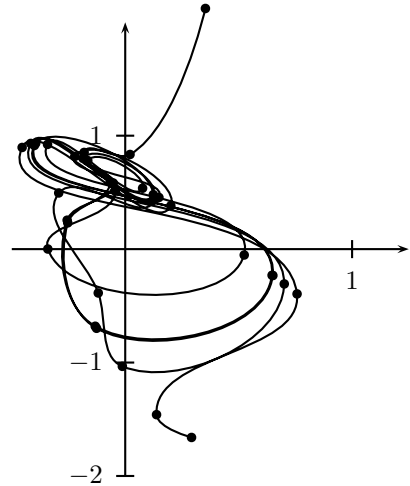
```
\newcommand{\DataB}{%
  [ ... Daten ... ]
  gsave           % speichere Grafikstatus
  /Helvetica findfont 40 scalefont setfont
  45 rotate       % Rotiere um 45 Grad
  0.9 setgray     % 1 entspricht Weiß
  -60 10 moveto (DRAFT) show
  grestore
}
\psset{xunit=1.5cm, yunit=3cm}
\begin{pspicture}(-3,-0.75)(2.25,1.5)
  \psaxes{->}(0,0)(-2,-0.5)(1.5,1.25)
  \listplot[%
    showpoints=true,%
    plotstyle=curve]{\dataA}
\end{pspicture}
\end{center}
```

`[ ... Daten ... ]` steht hierbei für die Liste aller $x|y$-Zahlenpaare, die hier aus Platzgründen nicht angegeben wurden.

Alternativ zum Manipulieren des Datensatzes von `\listplot` kann auch die entsprechende Funktion aus `pst-plot` verändert werden. Möchte man z.B. aus welchen Gründen auch immer die x/y-Werte vertauschen und den Graphen um 45° rotieren lassen (was einer Rotation mit anschließender Drehung entspricht), so kann dies einfach durch folgende Neudefinition von `\pst@def` erfolgen

```
\makeatletter
\pst@def{ScalePoints}<%
%-----------------------------------------------
  45 rotate % rotiere alle Objekte
%-----------------------------------------------
  /y ED /x ED
  counttomark dup dup cvi eq not { exch pop } if
  /m exch def /n m 2 div cvi def
  n {
%-----------------------------------------------
     exch % tausche letzen beiden Stackelemente
%-----------------------------------------------
     y mul m 1 roll
     x mul m 1 roll
     /m m 2 sub
     def } repeat>
\makeatother
```

Dies führt dann zur folgenden Abbildung:



Literatur

[1] Denis Girou: *Présentation de PSTricks*; *Cahier GUTenberg*; 16, S. 21–70; Apr. 1994.

[2] Michel Goosens, Frank Mittelbach und Alexander Samarin: *The LaTeX Graphics Companion*; Addison-Wesley Publishing Company; Reading, Mass.; 1997.

[3] Laura E. Jackson und Herbert Voß: *Die mathematischen Funktionen von Postscript*; *Die TeXnische Komödie*; 1/02; März 2002.

[4] Timothy van Zandt und Denis Girou: *Inside PSTricks*; *TUGboat*; 15, S. 239–246; Sept. 1994.

[5] Timothy Van Zandt: *PSTricks - PostScript macros for Generic TeX*; http://www.tug.org/application/PSTricks; 1993.