Kapitel 15

pst-plot: Plotten von Funktionen und Daten

Das Basispaket `pstricks` verfügt bereits über einige Makros, mit denen sich Funktionswerte bzw. Koordinatenpaare plotten lassen (siehe Abschnitt 15.4 auf Seite 152), wobei die Zahl der möglichen Koordinaten formal nicht begrenzt ist (Tabelle 15.1).

Tabelle 15.1: Zusammenstellung der Plotmakros des Basispakets `pstricks`

`\psdots`	→ 6.2 auf Seite 50
`\psline`	→ 4.2 auf Seite 30
`\pspolygon`	→ 4.4 auf Seite 31
`\pscurve`	→ 5.3.3 auf Seite 45
`\psecurve`	→ 5.3.4 auf Seite 45
`\psccurve`	→ 5.3.5 auf Seite 46

Das Paket `pst-plot` bietet darüber hinaus zum einen bessere Makros zum Plotten von externen Datensätzen und zum anderen Makros zum Plotten von Funktionen sowie einfaches Erstellen von Koordinatenachsen. [17] [18] [49] Grundsätzlich kann `pst-plot` nur zweidimensionale Datenpaare verarbeiten. Für das Darstellen von (x, y, z)-Datentripeln oder dreidimensionalen Funktionen kann auf das Paket `pst-3dplot` zurückgegriffen werden (→ 30 auf Seite 349). [44] [46]

15.1 Koordinatenachsen

Koordinatenachsen lassen sich prinzipiell auch mit den normalen Linienmakros zeichnen, jedoch vereinfacht sich das Problem bei Anwendung von `\psaxes` wesentlich (für die Option `{<Pfeile>}` siehe Tabelle 8.3 auf Seite 65).

```
\psaxes[<Optionen>]{<Pfeile>}(<x2,y2>)
\psaxes[<Optionen>]{<Pfeile>}(<x1,y1>)(<x2,y2>)
\psaxes[<Optionen>]{<Pfeile>}(<x0,y0>)(<x1,y1>)(<x2,y2>)
```

Sämtliche Koordinatenangaben müssen in kartesischer Form vorliegen, die Möglichkeit spezielle Angaben machen zu können (→ 12 auf Seite 107), ist hier nicht gegeben. Zwingend ist mindestens die Angabe von $(x2, y2)$. Abbildung 15.1 verdeutlicht die Zuordnung der einzelnen Punkte zu den gezeichneten Achsen.

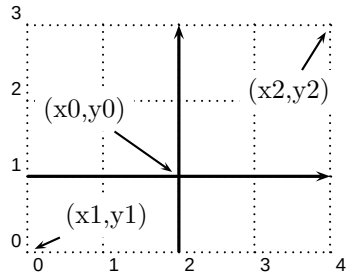


Abbildung 15.1: Bezugspunkte beim Zeichnen von Koordinatenachsen

Tabelle 15.2 zeigt eine Zusammenstellung der Parameter, die für das Erstellen von Koordinatenachsen von Interesse sind und in den folgenden Abschnitten eingehend erklärt werden. Hierbei hat der bereits früher definierte Parameter `labelsep` ebenfalls eine Bedeutung bei der Beschriftung der Achsen.

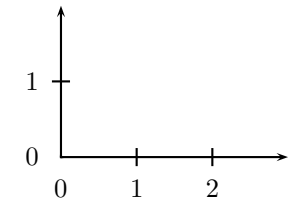
Tabelle 15.2: Zusammenfassung aller `psaxes` Parameter

Name	Werte	Vorgabe
axesstyle	axes|frame|none	axes
Ox	<Wert>	0
Oy	<Wert>	0
Dx	<Wert>	1
Dy	<Wert>	1
dx	<Wert [Einheit]>	0pt
dy	<Wert [Einheit]>	0pt
labels	all|x|y|none	all
showorigin	false|true	true
ticks	all|x|y|none	all
tickstyle	full|top|bottom	full
ticksize	<Wert [Einheit]>	3pt

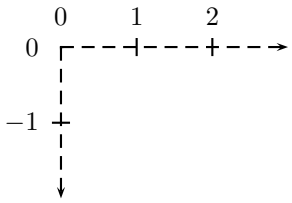
15.1.1 axesstyle

15.1.1.1 axes

Dieses entspricht der häufigsten Anwendung, es werden zwei Linien gezeichnet, wobei die Angabe des Koordinatenursprungs durch $(x0, y0)$ festgelegt wird. Die Platzierung der Label orientiert sich an der Anordnung der Achsen. Um die Beschriftung der Achsen nicht ausserhalb der eigentlichen `pspicture` Umgebung zu zeichnen, wird als unterer linker Punkt `(-0.5,-0.5)` gewählt.



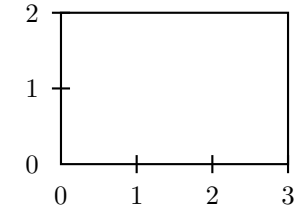

```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[axesstyle=axes]{->}(3,2)
\end{pspicture}
```



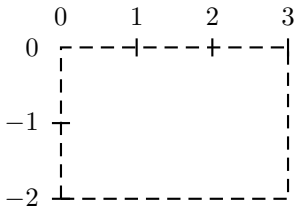

```
\begin{pspicture}(-0.5,0.5)(3,-2)
  \psaxes[axesstyle=axes,linestyle=dashed]{->}(3,-2)
\end{pspicture}
```

15.1.1.2 `frame`

Für diesen Fall macht es wenig Sinn, den Koordinatenursprung nicht in einer der Ecken anzuordnen.




```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[axesstyle=frame]{->}(3,2)
\end{pspicture}
```




```
\begin{pspicture}(-0.5,0.5)(3,-2)
  \psaxes[axesstyle=frame,linestyle=dashed]{->}(3,-2)
\end{pspicture}
```

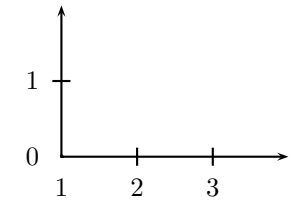
15.1.1.3 `none`

Dieser Fall erscheint formal sinnlos, dennoch bietet er die Möglichkeit, die Achsenlabels zu zeichnen, für die Linien aber eigene Vorstellungen zu verwirklichen.

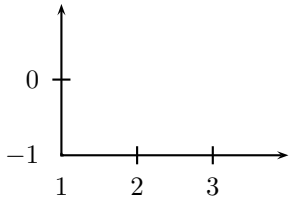

```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[axesstyle=none]{->}(3,2)
\end{pspicture}
```

15.1.2 0x und 0y

0x und 0x legen den Koordinatenursprung fest, der nach Voraussetzung bei $(0, 0)$ liegt. Grundsätzlich können hier beliebige reelle Zahlen angegeben werden, da PSTricks jedoch für die Label \multido (→ 33.3 auf Seite 433) benutzt, kann es zu falschen Ergebnissen kommen, denn \multido selbst hat nur eine rudimentäre Fließkomma-Arithmetik, die zu großen Ungenauigkeiten führen kann.



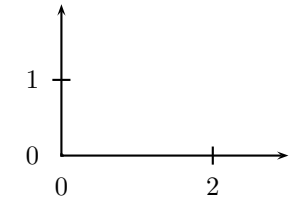

```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[0x=1]{->}(3,2)
\end{pspicture}
```



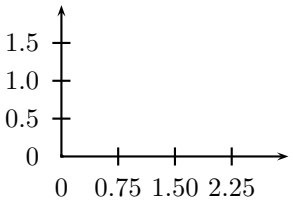

```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[0x=1,0y=-1]{->}(3,2)
\end{pspicture}
```

15.1.3 Dx und Dy

Dx und Dy bezeichnen den Abstand zweier aufeinanderfolgender Label. Da sich dieser Abstand auf den aktuellen Maßstab bezieht, wird hier nur eine Zahl angegeben. Grundsätzlich können hier reelle Zahlen angegeben werden, was insbesondere bei sehr kleinen Maßstäben der Fall sein wird.




```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[Dx=2]{->}(3,2)
\end{pspicture}
```



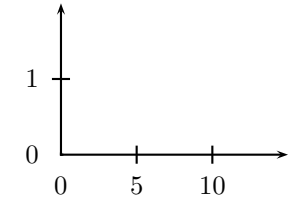

```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[Dx=0.75,Dy=0.5]{->}(3,2)
\end{pspicture}
```

15.1.4 dx und dy

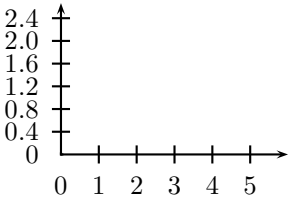
`dx` und `dy` bezeichnen den physischen Abstand zweier aufeinanderfolgender Label, müssen somit eine Einheit aufweisen. Wie man Tabelle 15.2 entnehmen kann, sind diese Werte mit `0pt` vorbesetzt, was auf den ersten Blick sinnlos erscheint. Intern wird aber für diesen Fall eine Ersetzung vorgenommen:

$$dx = 0 \rightarrow dx = Dx \cdot psxunit \tag{15.1}$$

$$dy = 0 \rightarrow dy = Dy \cdot psyunit \tag{15.2}$$




```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[Dx=5,dx=1]{->}(3,2)
\end{pspicture}
```




```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[Dx=1,dx=0.5,Dy=0.4,dy=0.3]{->}(3,2)
\end{pspicture}
```

15.1.5 labels

Mit `labels` kann man festlegen, welche der beiden Achsen eine Beschriftung erhalten. Der Abstand kann über den Parameter `labelsep` beeinflusst werden. Dagegen kann der Labelstil nur über die Neudefinition der beiden Label-Makros erfolgen.

```
\def\pshlabel#1{#1}
\def\psvlabel#1{#1}
```

Sollen beispielsweise die Label grundsätzlich nur in der Größe `\small` und im mathematischen Modusd gesetzt werden, so wird dieser Befehl einfach vor den Parameter gesetzt. Dies wurde für alle in diesem Kapitel angegeben Beispiele gemacht. Weitere Möglichkeiten zur Beeinflussung des Labelstils erhält man durch das Paket `pstricks-add` (→ 33.1 auf Seite 417).

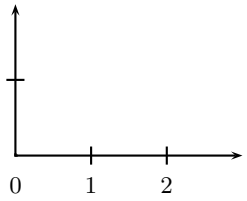
```
\def\pshlabel#1{\small $#1$}
\def\psvlabel#1{\small $#1$}
```

15.1.5.1 all

Dies entspricht der Vorgabe und ist in allen vorherigen Beispielen zu sehen.

15.1.5.2 x

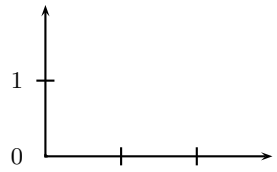
Nur die x-Achse bekommt Label.




```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[labels=x]{->}(3,2)
\end{pspicture}
```

15.1.5.3 y

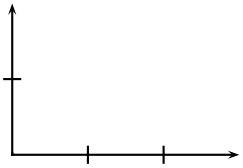
Nur die y-Achse bekommt Label.



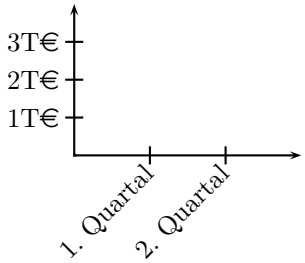

```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[labels=y]{->}(3,2)
\end{pspicture}
```

15.1.5.4 none

Es werden nur die Achsen ohne Label, aber mit den Ticks gezeichnet, was immer dann interessant ist, wenn die Beschreibung selbst vorgenommen werden soll.



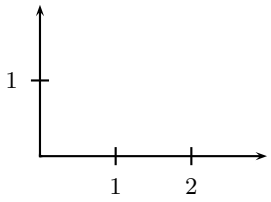
```
\begin{pspicture}(-1,-1)(3,2)
  \psaxes[labels=none]{->}(3,2)
\end{pspicture}
```




```
\begin{pspicture}(-1,-1)(3,2)
  \psaxes[labels=none,Dy=0.5]{->}(3,2)
  \rput[rC]{45}(1,-0.2){1. Quartal}
  \rput[rC]{45}(2,-0.2){2. Quartal}
  \rput[rC](-0.2,0.5){1T\euro}
  \rput[rC](-0.2,1){2T\euro}
  \rput[rC](-0.2,1.5){3T\euro}
\end{pspicture}
```

15.1.6 showorigin

Mit diesem Schalter lässt sich das Markieren des Ursprungs unterbinden, so fehlt im folgenden Beispiel das Label 0.




```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[showorigin=false]{->}(3,2)
\end{pspicture}
```

15.1.7 ticks

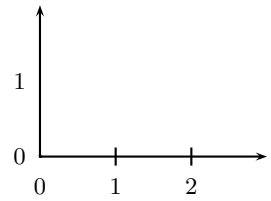
`ticks` bezeichnet, welche der Achsen Markierungen bekommen.

15.1.7.1 all

Dies entspricht der Vorgabe und ist in allen vorgehenden Beispielen zu sehen.

15.1.7.2 x

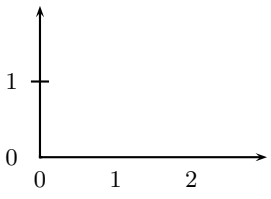
Nur die x-Achse bekommt Markierungen.




```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[ticks=x]{->}(3,2)
\end{pspicture}
```

15.1.7.3 y

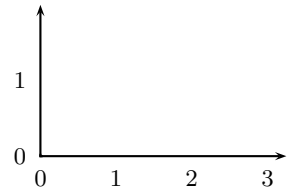
Nur die y-Achse bekommt Markierungen.



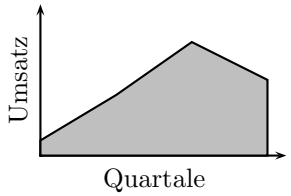

```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[ticks=y]{->}(3,2)
\end{pspicture}
```

15.1.7.4 none

Es werden nur die Achsen ohne Markierungen, aber mit den Label gezeichnet, was immer dann interessant ist, wenn die Grafik rein qualitativen Charakter hat.




```
\begin{pspicture}(-0.5,-0.5)(3.25,2)
  \psaxes[ticks=none]{->}(3.25,2)
\end{pspicture}
```




```
\begin{pspicture}(-0.5,-0.5)(3.25,2)
  \psaxes[ticks=none,labels=none]{->}(3.25,2)
  \uput[-90](1.5,0){Quartale}
  \uput[180]{90}(0,1){Umsatz}
  \pspolygon[fillcolor=lightgray,fillstyle=solid]%
     (0,0)(0,0.2)(1,0.8)(2,1.5)(3,1)(3,0)
\end{pspicture}
```

15.1.8 tickstyle

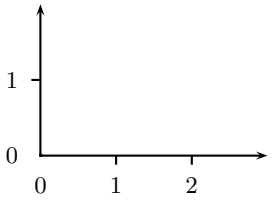
`tickstyle` gibt die Art der Markierungsstriche vor.

15.1.8.1 full

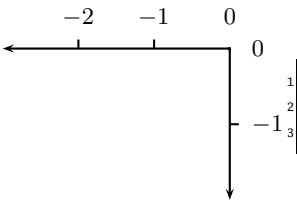
Dies entspricht der Vorgabe und ist in allen vorherigen Beispielen zu sehen.

15.1.8.2 bottom

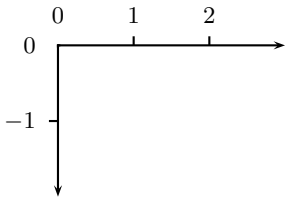
Die Striche werden für die y-Achse nur links und für die x-Achse nur unten gezeichnet. Dies kehrt sich um, wenn die Achsen in die negative Richtung zeigen.




```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[tickstyle=bottom]{->}(3,2)
\end{pspicture}
```



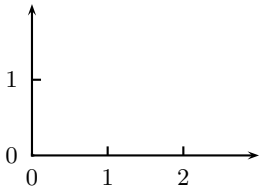

```
\begin{pspicture}(-3,0.5)(0.5,-2)
  \psaxes[tickstyle=bottom]{->}(-3,-2)
\end{pspicture}
```



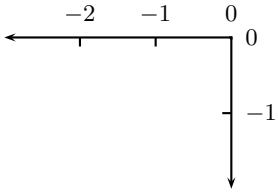

```
\begin{pspicture}(-0.5,0.5)(3,-2)
  \psaxes[tickstyle=bottom]{->}(3,-2)
\end{pspicture}
```

15.1.8.3 `top`

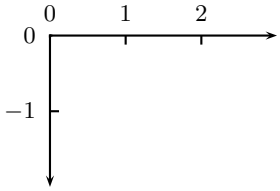
Die Striche werden für die y-Achse nur rechts und für die x-Achse nur oben gezeichnet. Dies kehrt sich um, wenn die Achsen in die negative Richtung zeigen.




```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[tickstyle=top]{->}(3,2)
\end{pspicture}
```



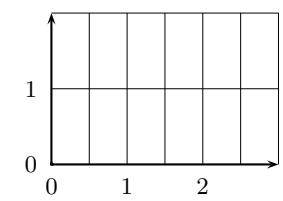

```
\begin{pspicture}(-3,0.5)(0.5,-2)
  \psaxes[tickstyle=top]{->}(-3,-2)
\end{pspicture}
```



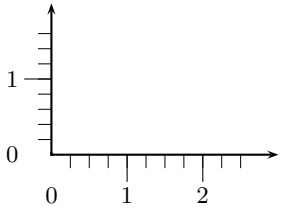

```
\begin{pspicture}(-0.5,0.5)(3,-2)
  \psaxes[tickstyle=top]{->}(3,-2)
\end{pspicture}
```

15.1.9 `ticksize`

`ticksize` gibt die halbe Länge der Markierungsstriche vor, womit sich auf leichte Weise der gesamte Achsenbereich markieren lässt, wenn man `tickstyle` entsprechend setzt. Die Größenangabe bezieht sich auf die aktuelle Maßeinheit.




```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[ticks=none]{->}(3,2)
  \psset{linewidth=0.1pt}%
  \psaxes[axesstyle=none,tickstyle=top,%
      ticksize=3,ticks=y,labels=none]{->}(3,2)
  \psaxes[axesstyle=none,tickstyle=top,ticksize=2,%
      ticks=x,Dx=0.5,labels=none]{->}(3,2)
\end{pspicture}
```




```
\begin{pspicture}(-0.5,-0.5)(3,2)
  \psaxes[ticks=none,labelsep=12pt]{->}(3,2)
  \psset{linewidth=0.1pt,axesstyle=none,%
      tickstyle=bottom,ticksize=5pt,labels=none}%
  \psaxes[ticks=x,Dx=0.25]{->}(2.5,1.75)
  \psaxes[ticks=y,Dy=0.2]{->}(2.5,1.75)
  \psset{linewidth=0.4pt,ticksize=10pt}%
  \psaxes[ticks=x]{->}(2.5,1.75)
  \psaxes[ticks=y]{->}(2.5,1.75)
\end{pspicture}
```

15.1.10 Erweiterungen

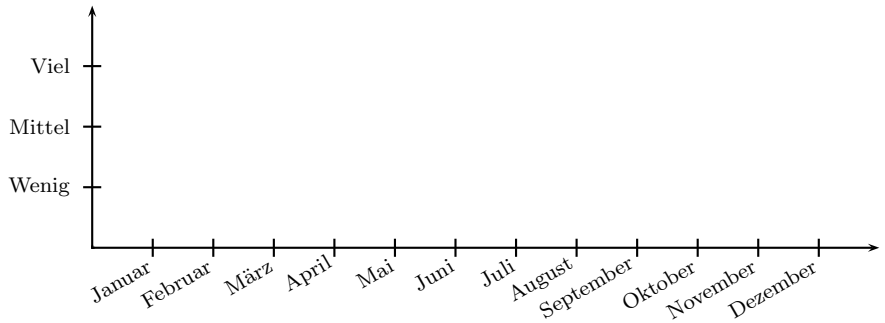
Häufig besteht der Wunsch, Achsen nicht mit Zahlenwerten, sondern irgendwelchen Symbolen oder Texten zu beschriften, bespielsweise den Monatsnamen. Das Beispiel in Abschnitt 15.1.5.4 auf Seite 136 zeigte bereits eine Möglichkeit, dies zu erreichen. Das Paket `arrayjob` kann hier eine weitere Untersützung liefern und die Beschriftung der Achsen erleichtern, denn es können beliebige alphanumerische Label definiert werden. [19]

Listing 15.1: Definition der Monatsnamen und relativer Werte

```
\def\Monat#1{%
  \ifcase#1\or
    Januar\or Februar\or März\or April\or Mai\or Juni\or%
    Juli\or August\or September\or Oktober\or November\or Dezember\fi
}%
\def\Level#1{%
  \ifcase#1\or Wenig\or Mittel\or Viel\fi%
}%
```

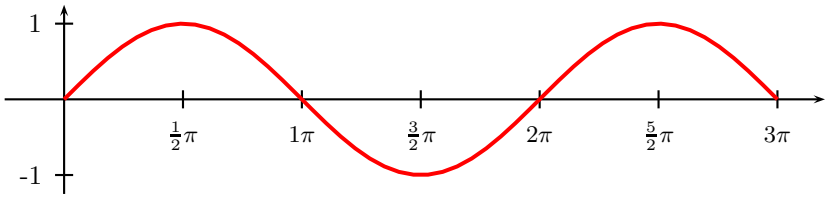
Andererseits kann man auch mit der `\ifcase` Anweisung arbeiten, die einem letztlich die gleichen Möglichkeiten gibt ohne ein externes Paket laden zu müssen. Benutzt werden weiterhin die Makros, die die Label setzen und von `PSTricks` als faktisch leer definiert sind und somit ganz einfach überschrieben werden können.

```
\def\psvlabel#1{#1}
\def\pshlabel#1{#1}
```




```
\def\psvlabel#1{\footnotesize\Level{#1}}
\def\pshlabel#1{\rput[rb]{30}{\footnotesize\Monat{#1}}}
\psset{unit=0.8}
\begin{pspicture}(-0.5,-1)(13,4)
  \psaxes[showorigin=false]{->}(13,4)
\end{pspicture}
```

Gleiches lässt sich mit Winkeleinheiten im Bogenmaß erreichen, wenn man die Möglichkeit der lokalen Änderung des Maßstabes berücksichtigt. Unter der Annahme, dass man eine Sinusfunktion für das Intervall $[0; 3\pi]$ zeichnen möchte, könnte man 6 Längeneinheiten für 3π nehmen, womit der Maßstabsfaktor dann $\frac{\pi}{2}$ beträgt. Die gesamte x-Achse hätte dann mindestens eine Länge von $6 \cdot \frac{\pi}{2} \approx 9.424777961cm$, wenn von der Einheit 1cm ausgegangen wird. Bei jeder Einheit muss auf der Achse ein Vielfaches von $\frac{\pi}{2}$ markiert werden. Mit der durch `pstricks-add` (→ 33.1 auf Seite 417) definierten Modulo-Funktion kann man mit ein wenig TEX Kenntnissen die Achse sinnvoll beschriften. `\psplot` wird mit dem normalen Maßstab aufgerufen, sodass man einfach das Intervall $[0; 2\pi]$ für die Funktion benutzen kann.




```
\makeatletter
\def\pshlabel#1{\small%
   \pst@mod{#1}{2}\tempa%     0 oder 1
   \ifnum1>\tempa%            ungerade?
     \count1=#1\divide\count1 by 2% #1/2
      $\the\count1\pi$%       n*pi
   \else$\frac{#1}{2}\pi$\fi% n/2*pi
\makeatother
\begin{pspicture}(-0.5,-1.25)(10,1.25)
  \psaxes[xunit=1.570796327,showorigin=false]{->}(0,0)(-0.5,-1.25)(6.4,1.25)
  \psplot[linecolor=red,linewidth=1.5pt]{0}{9.424777961}{x 180 mul 3.141592654 div
     sin}
\end{pspicture}
```

15.2 Parameter

Tabelle 15.3 zeigt eine Zusammenstellung der speziellen für `pst-plot` geltenden Parameter.

Tabelle 15.3: Zusammenfassung aller Parameter für `pst-plot`

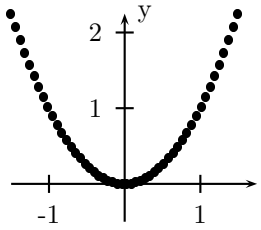
Name	Werte	Vorgabe
`plotstyle`	`dots|line|polygon|curve|ecurve|ccurve`	`line`
`plotpoints`	`<Wert>`	50

15.2.1 `plotstyle`

Wie in den folgenden Beispielen zu sehen ist, unterscheiden sich die Darstellungen für die Plotstile `curve`, `ecurve` und `ccurve` faktisch nicht. Dies ist in der Regel immer der Fall, wenn mathematische Funktionen zugrunde liegen.

15.2.1.1 `dots`

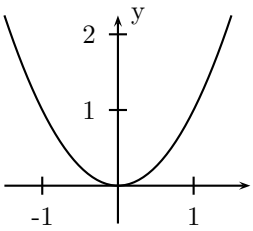
Es werden lediglich die Koordinaten durch Punkte markiert, deren Aussehen durch die Parameter aus Tabelle 6.1 auf Seite 47 beeinflusst werden kann. Allerdings können hier auch eigene Symbole definiert werden (→ 6.3 auf Seite 50).




```
\begin{pspicture}(-1.75,-0.5)(1.75,2.3)
 \psaxes{->}(0,0)(-1.5,-0.5)(1.75,2.25)
 \uput[-90](1,75,0){x}
 \uput[0](0,2.25){y}
 \psplot[plotstyle=dots]{-1.5}{1.5}{x dup mul}
\end{pspicture}
```

15.2.1.2 `line`

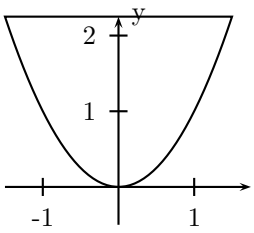
Es werden die Koordinaten durch Sekanten (Linien) verbunden, deren Aussehen durch die Parameter aus Tabelle 4.1 auf Seite 23 beeinflusst werden kann.




```
\begin{pspicture}(-1.75,-0.5)(1.75,2.3)
 \psaxes{->}(0,0)(-1.5,-0.5)(1.75,2.25)
 \uput[-90](1,75,0){x}
 \uput[0](0,2.25){y}
 \psplot[plotstyle=line]{-1.5}{1.5}{x dup mul}
\end{pspicture}
```

15.2.1.3 polygon

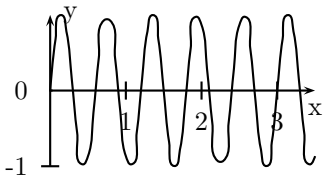
Mit dieser Option erreicht man das gleiche Verhalten wie mit dem Makro `\pspolygon` (→ 4.4 auf Seite 31). Dies bedeutet, dass die Kurve am Ende geschlossen wird, indem der Endpunkt auf den Anfangspunkt zurückgeführt wird.



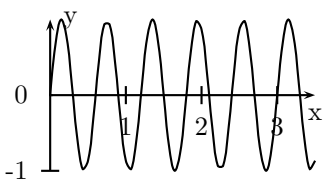

```
\begin{pspicture}(-1.75,-0.5)(1.75,2.3)
  \psaxes{->}(0,0)(-1.5,-0.5)(1.75,2.25)
  \uput[-90](1,75,0){x}
  \uput[0](0,2.25){y}
  \psplot[plotstyle=polygon]{-1.5}{1.5}{x dup mul}
\end{pspicture}
```

15.2.1.4 curve

Insbesondere bei sehr steilen Kurven kann es mit der `curve` Option zu Schwierigkeiten kommen. Eventuell muss über den Parameter `curvature` (→ 5.1.2 auf Seite 36) eine Anpassung vorgenommen werden. Die Bedeutung von `curve` ist in Abschnitt 5.3.3 auf Seite 45 erläutert.



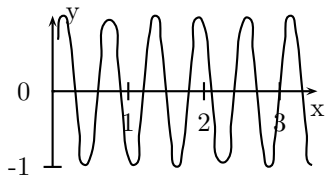

```
\begin{pspicture}(0,-1)(3.5,1)
  \psaxes{->}(0,0)(0,-1)(3.5,1)
  \uput[-90](3.5,0){x}
  \uput[0](0,1){y}
  \psplot[plotstyle=curve]{0}{3.5}{x 360 mul 0.6 div sin}
\end{pspicture}
```



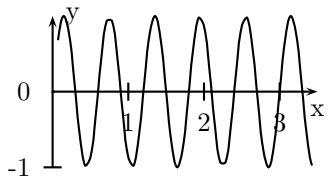

```
\begin{pspicture}(0,-1)(3.5,1)
  \psaxes{->}(0,0)(0,-1)(3.5,1)
  \uput[-90](3.5,0){x}
  \uput[0](0,1){y}
  \psplot[plotstyle=curve,curvature=1 1 -1]%
      {0}{3.5}{x 360 mul 0.6 div sin}
\end{pspicture}
```

15.2.1.5 ecurve

Auch hier kann es bei sehr steilen Kurven zu Schwierigkeiten kommen. Eventuell muss über den Parameter `curvature` (→ 5.1.2 auf Seite 36) eine Anpassung vorgenommen werden. Die Bedeutung von `ecurve` ist in Abschnitt 5.3.4 auf Seite 45 erläutert.



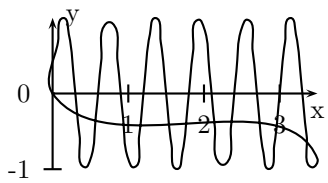

```
\begin{pspicture}(0,-1)(3.5,1)
  \psaxes{->}(0,0)(0,-1)(3.5,1)
  \uput[-90](3.5,0){x}
  \uput[0](0,1){y}
  \psplot[plotstyle=ecurve]{0}{3.5}%
       {x 360 mul 0.6 div sin}
\end{pspicture}
```



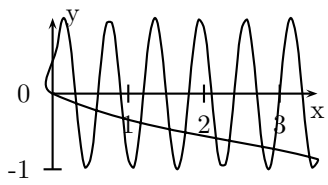

```
\begin{pspicture}(0,-1)(3.5,1)
  \psaxes{->}(0,0)(0,-1)(3.5,1)
  \uput[-90](3.5,0){x}
  \uput[0](0,1){y}
  \psplot[plotstyle=ecurve,curvature=1 1 -1]%
       {0}{3.5}{x 360 mul 0.6 div sin}
\end{pspicture}
```

15.2.1.6 ccurve

Auch die `ccurve` Option kann bei sehr steilen Kurven problematisch sein. Eventuell muss über den Parameter `curvature` (→ 5.1.2 auf Seite 36) eine Anpassung vorgenommen werden. Die Bedeutung von `ccurve` ist in Abschnitt 5.3.5 auf Seite 46 erläutert.



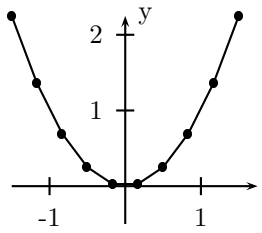

```
\begin{pspicture}(0,-1)(3.5,1)
  \psaxes{->}(0,0)(0,-1)(3.5,1)
  \uput[-90](3.5,0){x}
  \uput[0](0,1){y}
  \psplot[plotstyle=ccurve]{0}{3.5}%
      {x 360 mul 0.6 div sin}
\end{pspicture}
```




```
\begin{pspicture}(0,-1)(3.5,1)
  \psaxes{->}(0,0)(0,-1)(3.5,1)
  \uput[-90](3.5,0){x}
  \uput[0](0,1){y}
  \psplot[plotstyle=ccurve,curvature=1 1 -1]%
       {0}{3.5}{x 360 mul 0.6 div sin}
\end{pspicture}
```

15.2.2 plotpoints

Dieser Parameter beeinflusst wesentlich die Darstellung von Kurven. Die Vorgabe von 50 Punkten für das angegebene Intervall ist sicherlich für viele Funktionen ausreichend, aber sehr oft auch zu wenig. Bei den heutigen Rechnerleistungen können hier durchaus Werte von 5000 und mehr angegeben werden. Andererseits kann man bei Funktionen mit geringen Steigungsunterschieden auch sehr wenige Punkte wählen. Dabei muss eventuell die Druckerauflösung beachtet werden.




```
\begin{pspicture}(-1.75,-0.5)(1.75,2.3)
 \psaxes{->}(0,0)(-1.5,-0.5)(1.75,2.25)
 \uput[-90](1,75,0){x}
 \uput[0](0,2.25){y}
 \psplot[plotpoints=10,showpoints=true]{-1.5}{1.5}{x
    dup mul}
\end{pspicture}
```

showpoints → 4.1 auf Seite 23

15.3 Plotten von Daten

pst-plot Das Paket stellt die folgenden drei Plotfunktionen zur Verfügung:

```
\fileplot[<Parameter>]{<Dateiname>}
\fileplot*[<Parameter>]{<Dateiname>}
\dataplot[<Parameter>]{<Makroname>}
\dataplot*[<Parameter>]{<Makroname>}
\listplot[<Parameter>]{<Werteiste>}
\listplot*[<Parameter>]{<Werteiste>}
\readdata{<Makroname>}{<Dateiname>}
\savedata{<Makroname>}[<Werteliste>]
```

Die Sternversionen sind prinzipiell nicht weiter von Interesse, denn im allgemeinen ist man nicht daran interssiert, dass die Folge von Daten zu einer geschlossenen Kurve erweitert und mit der aktuellen Linienfarbe gefüllt wird.

15.3.1 Datenstruktur

Die externen oder als Zahlenpaare übergebenen Daten sind als reine Zahlenwerte paarweise 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}
```

Alle Daten mit eckigen Klammern[1] zu versehen, beschleunigt erheblich die Leserate, denn PostScript kann die Daten dann als array einlesen und entsprechend schneller verarbeiten. Andererseits gelten hier gerätespezifische Begrenzungen, wieviel Daten TEX in einem Durchgang einlesen kann. Eine andere Möglichkeit,

[1] „[“ muss am Anfang einer Zeile stehen.

Speicherproblemen aus dem Weg zu gehen, ist die Anwendung des PSTtoEPS-Makros (→ 21 auf Seite 245).

Der für Daten als Trenner häufig benutzte Tabulator (\t) bzw. \009) ist nicht zulässig, kann aber leicht für Unix(e) ersetzt werden.

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

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

15.3.2 \readdata und \savedata

Die Anwendung dieser beiden Makros ist denkbar einfach, denn als Parameter wird bei \readdata ein Makroname und ein Dateiname erwartet. Bei \savedata ist es ein Makroname und eine Werteliste. Wird kein absoluter oder relativer Pfad beim Dateinamen angegeben, so gilt immer das aktuelle Verzeichnis des Dokuments.

In manchen Anwendungsfällen sind Datenpaare jeweils mit ihren Fehlern erfasst. Dabei wäre es angenehm, wenn man dies optisch neben dem eigentlichen Datum dargestellen könnte. Normalerweise kann PSTricks Daten nur als Zahlenpaar oder Zahlentripel (→ 30 auf Seite 349) einlesen. Benutzt man das Makro \readdata (→ 15.3.4 auf Seite 148), so lässt sich erst einmal **jede** Liste an Daten einlesen, denn sie werden nur in folgender Form im angegebenen Makro gespeichert:

```
D <Wert1> D <Wert2> D <Wert3> ...
```

Damit kann man sie natürlich beliebig manipulieren, zumal man mit \@ifnextchar D jederzeit feststellen kann, ob noch Daten vorhanden sind. Für das Beispiel liegen folgende Daten zugrunde, die alle den Aufbau x y dmin dmax haben:

Listing 15.2: Inhalt der Datendatei dataError.dat

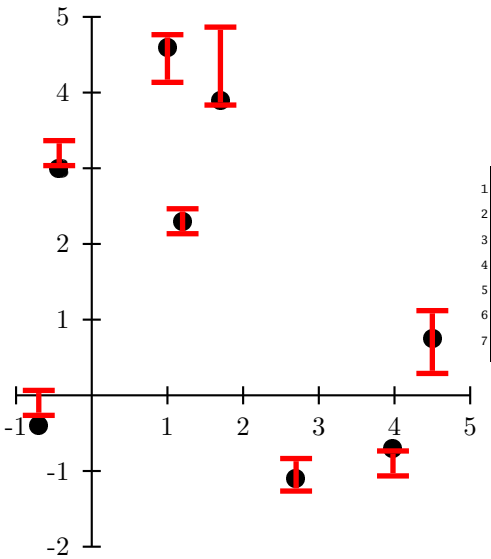
```
1 -0.7 -0.4      0.1 0.5       -0.43 3         0   0.4
2  1    4.6     -0.5 0.2        1.2  2.3     -0.2 0.2
3  1.7  3.9     -0.1 1          2.7 -1.1     -0.2 0.3
4  3.98 -0.7    -0.4 0          4.5  0.7539  -0.5 0.4
```

dmax gibt die maximale Messwertabweichung nach oben und dmin nach unten an, jeweils in demselben Maßstab wie die Messwerte. Nach dem Einlesen mit \readdata{\Data}{dataError.dat} enthält das Makro \Data den gesamten Datensatz der Datei dataError.dat in der Form:

```
D -0.7 D -0.4 D 0.1 D 0.5 D -0.43 D 3 D 0 D 0.4 D 1 D 4.6 D -0.5 D 0.2
D 1.2 D 2.3 D -0.2 D 0.2 D 1.7 D 3.9 D -0.1 D 1 D 2.7 D -1.1 D -0.2 D 0.3
D 3.98 D -0.7 D -0.4 D 0 D 4.5 D 0.7539 D -0.5 D 0.4
```

der durch \show\Data automatisch zur Kontrolle ins Logfile geschrieben werden kann.

Statt eines einzigen Punktes ist jetzt noch zusätzlich eine vertikale Linie in der Form `\psline{|-|}(x,y+dmax)(x,y+dmin)` zu zeichnen. Das `D` in den Datensätzen lässt sich leicht herausfiltern, wenn man insgesamt acht Variablen aus dem Makro `\Data` einliest, aber nur jede zweite berücksichtigt. Die Koordinaten für den Fehlerbalken kann man leicht mithilfe der speziellen PostScript-Koordinaten berechnen. Die entsprechende Ausgabe zeigt das folgende Beispiel.




```
\readdata{\Data}{dataError.dat}
\psset{dotscale=2}%
\begin{pspicture}(-1,-2)(5,5.5)
  \psaxes(0,0)(-1,-2)(5,5)
  \def\DoCoordinate#1#2{\psdot(#1,#2)}%
  \GetCoordinates{\Data}
\end{pspicture}
```

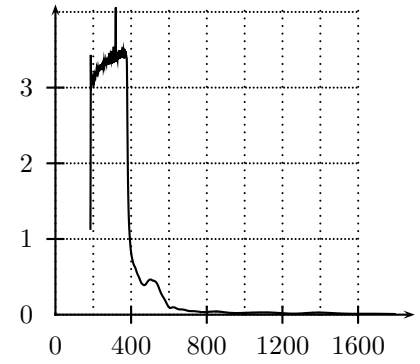
```
\bgroup
\makeatletter
\def\errorLine{\@ifnextchar[{\pst@errorLine}{\pst@errorLine[]}}
\def\pst@errorLine[#1](#2)#3#4{{%
  \ifx#1\empty\else\psset{#1}\fi
  \pst@getcoor{#2}\pst@tempa
  \psline{|-|}(!%
     /yDot \pst@tempa exch pop \pst@number\psyunit div def
     /xDot \pst@tempa pop \pst@number\psxunit div def
     xDot yDot #3\space add%
    )(!%
     /yDot \pst@tempa exch pop \pst@number\psyunit div def
     /xDot \pst@tempa pop \pst@number\psxunit div def
     xDot yDot #4\space add)
}}
%
\def\GetCoordinates#1{\expandafter\GetCoordinates@i#1}
\def\GetCoordinates@i #1{\GetCoordinates@ii#1}
\def\GetCoordinates@ii#1 #2 #3 #4 #5 #6 #7 #8 {%
  \DoCoordinate{#2}{#4}%
  \errorLine[linecolor=red, linewidth=2pt](#2,#4){#6}{#8}% <<<<<
  \@ifnextchar D{\GetCoordinates@ii}{}%
}
\makeatother
\egroup
```

15.3.3 \fileplot

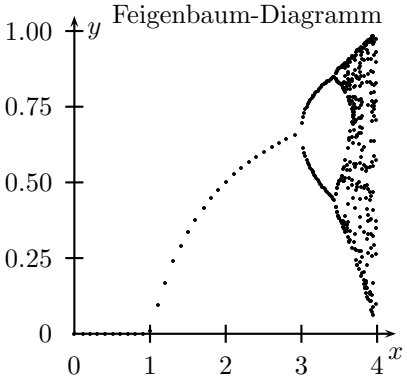
\fileplot ist das am einfachsten anzuwendende Makro und ist immer dann angebracht, wenn in einer externen Datei gespeicherte Zahlenpaare $(x|y)$ geplottet werden sollen. Auf der anderen Seite weist fileplot einige Einschränkungen auf.

Es sind keine „curve“ Plotstile möglich und Parameterangaben zu arrows, linearc und showpoints werden ignoriert.

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=0.025mm}
\begin{pspicture}(-200,-0.5)(1900,4.25)
  \fileplot[plotstyle=line]{fileplot.data}
  \psaxes[dx=400,Dx=400]{->}(1900,4.1)
  \psgrid[griddots=10,subgriddiv=0,%
      xunit=0.5cm,gridlabels=0pt](8,4)
\end{pspicture}
```



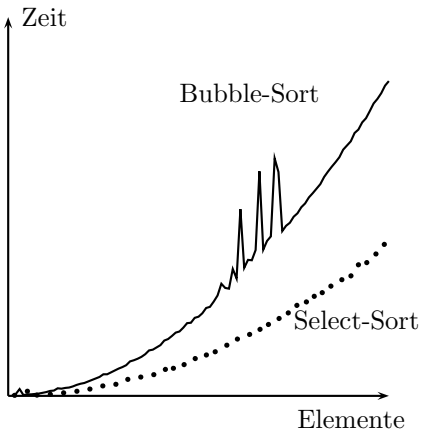

```
\psset{yunit=4cm}
\begin{pspicture}(-0.75,-0.1)(4.25,1.05)
  \fileplot[plotstyle=dots,dotsize=1.5pt]%
     {feigenbaum.data}
  \psaxes[Dy=0.25]{->}(4.25,1.05)
  \uput[-90](4.25,0){$x$}
  \uput[0](0,1){$y$}
  \rput[l](0.5,1.05){Feigenbaum-Diagramm}
\end{pspicture}
```

15.3.4 \dataplot

Ebenso wie \fileplot benötigt \dataplot einen externen Datensatz, der jedoch nicht als Datei, sondern als Makro vorliegt. Das Einlesen einer externen Datei und Abspeichern in einem Makro kann allerdings von \readdata übernommen werden, beispielsweise:

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

Die Zahl der eingelesenen Dateien bzw. Daten, ist nur durch den Speicher bestimmt. Mit `\dataplot` sind einfache Overlays möglich. Das angegebene Beispiel zeigt zwei getrennte Datendateien, die in einem Koordinatensystem dargestellt werden.




```
\psset{xunit=0.0005cm,yunit=0.005cm}
\begin{pspicture}(0,-50)(10000,1100)
  \readdata{\bubble}{bubble.data}
  \readdata{\select}{select.data}
  \dataplot[plotstyle=line]{\bubble}
  \dataplot[plotstyle=line,%
      linestyle=dotted,%
      linewidth=2pt]{\select}
  \psline{->}(0,0)(10000,0)
  \uput[-90](9000,0){Elemente}
  \psline{->}(0,0)(0,1000)
  \uput[0](0,1000){Zeit}
  \rput[l](4500,800){Bubble-Sort}
  \rput[l](7500,200){Select-Sort}
\end{pspicture}
```

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

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

15.3.5 `\listplot`

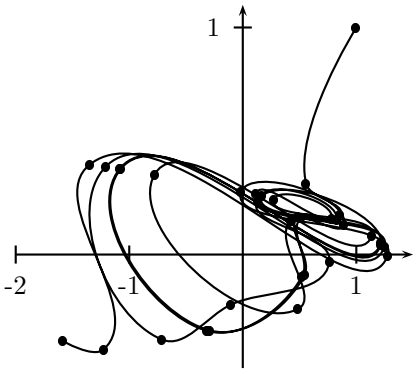
Im Gegensatz zu den vorhergehenden Plotmakros wird das Argument von `\listplot` zuerst von TEX expandiert, wenn es sich um ein TEX-Befehl handelt, andernfalls wird es unverändert nach PostScript durchgereicht, dabei werden aber TEX-spezifische Makros durch ihr Argument ersetzt. Daraus folgt, dass man komplette PostScript-Programme im Argument von `\listplot` ablegen kann.

Das angegebene Beispiel zeigt in der Originaldarstellung die Entwicklung eines Henon-Attraktors. Die untere Grafik enthält zusätzlich zum normalen Datensatz einen durch zusätzlichen PostScript-Code erzwungenen „Draft“-Hinweis.

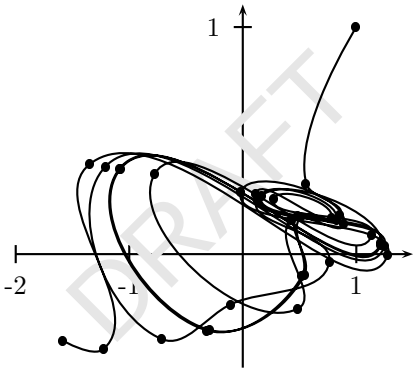
Listing 15.3: Einfügen eines „DRAFT“ Hinweises

```
\newcommand{\DataB}{%
  [ ... data ... ]
  gsave          % save graphics state
  /Helvetica findfont 40 scalefont setfont
```

```
  45 rotate      % rotating angle
  0.9 setgray    % 1 is white
  -60 10 moveto (DRAFT) show
  grestore
}
```




```
\psset{xunit=1.5cm, yunit=3cm}
\begin{pspicture}(-2,-0.5)(2.25,1.1)
  \psaxes{->}(0,0)(-2,-0.5)(1.5,1.1)
  \listplot[showpoints=true,plotstyle=curve
    ]{\henon}
\end{pspicture}
```




```
\psset{xunit=1.5cm, yunit=3cm}
\begin{pspicture}(-2,-0.5)(2.25,1.1)
  \psaxes{->}(0,0)(-2,-0.5)(1.5,1.1)
  \listplot[showpoints=true,%
        plotstyle=curve]{\dataA}
\end{pspicture}
```

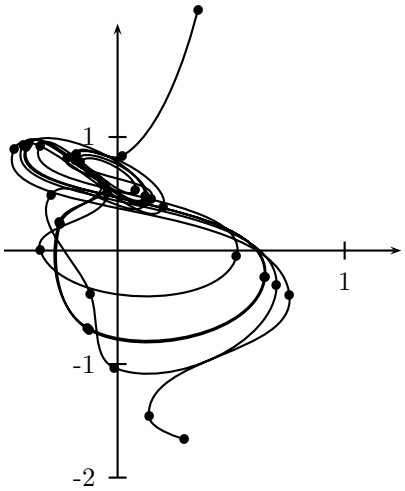
Alternativ zum Manipulieren des Datensatzes von `\listplot` kann auch die entsprechende Funktion aus `pst-plot` verändert werden. Möchte man beispielsweise, aus welchen Gründen auch immer, die $x|y$-Werte vertauschen und den Graphen um 45° rotieren lassen (was einer Rotation mit anschließender Drehung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

Listing 15.4: Drehung des Koordinatensystems

```
\makeatletter
\pst@def{ScalePoints}<%
%-----------------------------------------------
  45 rotate % rotate all objects
%-----------------------------------------------
  /y ED /x ED
  counttomark dup dup cvi eq not { exch pop } if
  /m exch def /n m 2 div cvi def
  n {
%-----------------------------------------------
```

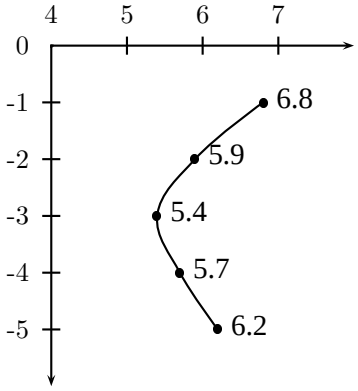
```
     exch % exchanges the last two stackelements
%-----------------------------------------------
     y mul m 1 roll
     x mul m 1 roll
     /m m 2 sub
     def } repeat>
\makeatother
```

Dies führt dann zur folgenden Abbildung.




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

Mit `\pscustom` und dem PostScript-nahen Makro `\code` kann man faktisch jede beliebige Manipulation auf PostScript-Ebene durchführen, ohne dabei selbst in das `\listplot`-Makro einzugreifen. Im folgenden Beispiel werden nach dem Zeichnen der Daten zusätzlich die Koordinaten an die Punkte gesetzt.




```
\makeatletter
\psset{yunit=1cm,xunit=0.75cm}
\def\plotValues#1{%
  \pscustom{%
```

```
    \code{%
      /xOffset 5 def  /yOffset -2 def
      /Times findfont 11 scalefont setfont
      /Feld [ #1 ] def
      /cnt 0 def
      Feld length 2 div cvi {
        /x Feld cnt get def
        /y Feld cnt 1 add get def
        x \pst@number\psxunit mul xOffset add
        y \pst@number\psyunit mul yOffset add
        moveto x 10 string cvs show
        /cnt cnt 2 add def
      } repeat
}}}
\makeatother
\begin{pspicture}(3.5,0.5)(8,-5.5)
  \psaxes[0x=4]{->}(4,0)(8,-5.5)
  \def\data{6.8 -1 5.9 -2 5.4 -3 5.7 -4 6.2 -5}
  \listplot[plotstyle=curve, showpoints=true]{\data}
  \plotValues{\data}
\end{pspicture}
```

15.4 Plotten von Funktionen

PostScript arbeitet mit dem so genannten Stacksystem, welches den Benutzern von HP-Taschenrechnern geläufig und auch unter dem Namen **UPN**, *Umgekehrte Polnische Notation* (Reverse Polish Notation), bekannt ist und letztlich den internen Standard für alle Computer darstellt. Die normale Notation für die Multiplikation „$a * b =$“ wird zu „a<enter>b<enter>$*$“. Es sind immer zuerst die Parameter (Variablen) auf dem Stack abzulegen (durch `<enter>` symbolisiert), bevor eine der mathematischen Funktionen aufgerufen wird. Die hier beschriebenen Befehle beziehen sich immer auf das oberste Stackelement oder die obersten beiden Stackelemente. Grundsätzlich kann man bei Problemen aber einen sogenannten `Infix-Postfix`-Konverter verwenden, der „normale“ (Infix) mathematische Ausdrücke in solche mit UPN-Notation (Postfix) wandelt. [34]

Die direkte Anwendung der PostScript-Befehle für die Darstellung mathematischer Zusammenhänge bringt gegenüber Programmen wie beispielsweise `gnuplot` nicht unbedingt Vorteile in der endgültigen Druckausgabe. Auch lässt sich nicht immer jedes mathematische Problem mit den PostScript-Befehlen einfach lösen.

Die grundsätzliche Struktur der hier behandelten Makros ist:

```
\psplot[<Optionen>]{<xmin>}{<xmax>}{<Funktion f(x)>}
\parametricplot[<Optionen>]{<tmin>}{<tmax>}{<Funktionen x(t) y(t)>}
```

Hierin bedeuten $[x_{min};\ x_{max}]$ bzw. $[t_{min};\ t_{max}]$ das jeweilige Definitionsintervall (Start- und Endwert). Die möglichen Parameter sind in Tabelle 15.3 auf Seite 142 zusammengefasst, wobei im Zusammenhang mit den Funktionen lediglich der Parameter `plotpoints` interessant ist, welcher die Anzahl der Stützstellen angibt

und standardmäßig auf 50 gesetzt ist. Im Normalfall sollten sämtliche berechneten Punkte mit dem `plotstyle=lines` verbunden werden, sodass sich eine zu geringe Anzahl an Stützstellen durch einen Polygonzug bemerkbar macht. Mit Werten um 200 liegt man in den meisten Fällen auf der richtigen Seite.

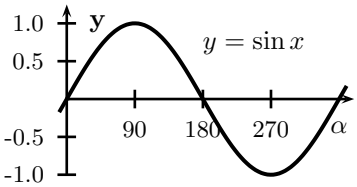
Der Variablenname für `\psplot` ist per Definition x und für `\parametricplot` t. Beide können nicht ohne weiteres verändert werden, was jedoch bezüglich der Anwendung keinerlei Einschränkung darstellt. Die Variablen können beliebig oft innerhalb eines Ausdrucks verwendet werden, denn erst mit der schließenden Klammer für den Funktionsausdruck wird davon ausgegangen, dass die zweite Koordinate für einen Punkt des Graphen oben auf dem Stack liegt. Der einzige Unterschied zwischen diesen beiden Befehlen ist, dass bei `psplot` nur der oberste Stackwert (y) und bei `parametricplot` die obersten beiden Stackwerte ($x; y$) als Argumente verwendet werden.

Zu beachten ist unbedingt, dass mit den beiden Befehlen keine Fehlermeldungen ausgegeben werden, was insbesondere für diejenigen mathematischen Funktionen von Interesse ist, deren Definitionsbereich nicht gleich dem der reellen Zahlen entspricht. Denn bei **einem** fehlerhaften Argument, z.B. $\sqrt{-1}$, wird der komplette Graph nicht gezeichnet!

15.4.1 \psplot

15.4.1.1 Sinusfunktion

Funktion	PostScript
$y(x) = \sin x$	x sin

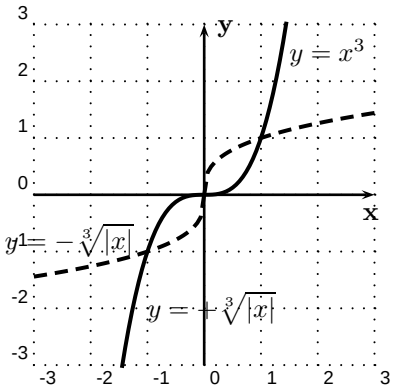



```
\psset{xunit=0.01cm,yunit=1cm}
\begin{pspicture}(-80,-1.25)(400,1.25)%
  \psaxes[linewidth=1pt,showorigin=false,%
      Dx=90,Dy=0.5]{->}(0,0)(0,-1)(380,1.25)
  \uput{0.3}[-90](360,0){$\mathbf{\alpha}$}
  \uput{0.3}[0](0,1){$\mathbf{y}$}
  \psplot[plotstyle=curve,%
      linewidth=1.5pt]{-10}{370}{x sin}
  \rput[l](180,0.75){$y=\sin x$}%
\end{pspicture}
```

15.4.1.2 Potenzfunktion

Dargestellt wird eine Parabel dritten Grades sowie ihre Umkehrfunktion, wobei die Intervallunterscheidung nicht zwingend ist, wenn man die Exponentialschreibweise mit $y = x^{-\frac{1}{3}}$ wählt.

Funktion	PostScript
$y(x) = x^3$	x 3 exp
$y^{-1}(x) = \begin{cases} +\sqrt[3]{\lvert x\rvert} & x > 0 \\ -\sqrt[3]{\lvert x\rvert} & x < 0 \end{cases}$	x 0.333 exp (Umkehrfunktion)

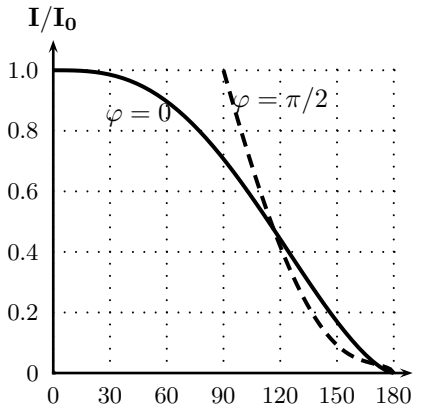



```
\psset{unit=0.75cm}
\begin{pspicture}(-3.25,-3)(3.25,3)%
  \psgrid[subgriddiv=0,griddots=5,gridlabels=7
    pt]
  \psaxes[linewidth=1pt,ticks=none,%
    labels=none]{->}(0,0)(-3,-3)(3,3)
  \uput[-100](3,0){$\mathbf{x}$}
  \uput[-10](0,3){$\mathbf{y}$}
  \psset{linewidth=1.5pt}
  \psplot{-1.45}{1.45}{x 3 exp}
  \psset{linestyle=dashed}
  \psplot{0}{3}{x 0.333 exp}
  \psplot{-3}{0}{x -1 mul 0.333 exp -1 mul}
  \rput[l](1.5,2.5){$y=x^3$}
  \rput[l](-1,-2){$y=+\sqrt[3]{|x|}$}
  \rput[r](-1.25,-0.8){$y=-\sqrt[3]{|x|}$}
\end{pspicture}
```

15.4.1.3 Beispiel aus der Leistungselektronik

Gezeigt wird die grafische Darstellung des relativen Strommittelwertes für eine Stromrichtersteuerung durch ein Thyristorpaar. Hierbei entspricht der Parameter φ der Phasenverschiebung zwischen Strom und Spannung. Die unabhängige Variable α bezeichnet den Steuerwinkel.

Funktion	$\frac{I(\alpha)}{I_0} = \begin{cases} \sqrt{1 - \frac{\alpha}{\pi} + \frac{1}{2\pi}\sin 2\alpha} & \varphi = 0 \\ \sqrt{\left(2 - \frac{2\alpha}{\pi}\right)(2 + \cos 2\alpha) + \frac{3}{\pi}\sin 2\alpha} & \varphi = \frac{\pi}{2} \end{cases}$	
PostScript	1 x 180 div sub 1 6.28 div x 2 mul sin mul add sqrt 2 x 90 div sub x 2 mul cos 2 add mul x 2 mul sin 3 3.15 div mul add sqrt	$\varphi = 0$ $\varphi = \frac{\pi}{2}$




```
\psset{xunit=0.025cm,yunit=4cm}
\begin{pspicture}(-0.1,-0.1)(190,1.1)
  \psgrid[subgriddiv=0,griddots=5,%
    gridlabels=0pt,xunit=30,yunit=0.2](6,5)
  \psaxes[linewidth=1pt,ticks=none,Dx=30,%
    Dy=0.2]{->}(190,1.1)
  \uput{0.5}[0](180,0){$\mathbf{\alpha}$}
  \uput{0.5}[90](0,1){$\mathbf{I/I_0}$}
  \psset{linewidth=1.5pt}
  \psplot{0}{180}{1 x 180 div sub 1
   6.28 div x 2 mul sin mul add abs sqrt}
  \psplot[linestyle=dashed]{90}{180}{%
   2 x 90 div sub x 2 mul cos 2 add mul x
   2 mul sin 3 3.15 div mul add abs sqrt}
  \rput(45,0.85){$\varphi=0$}
  \rput(120,0.9){$\varphi=\pi/2$}
\end{pspicture}
```

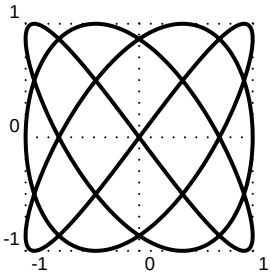
Zu beachten ist, dass PostScript die Argumente für die trigonometrischen Funktionen im Gradmaß erwartet, sodass für relative Winkel auf gleiche Einheiten zu achten ist. Der Ausdruck $\frac{\alpha}{\pi}$ ist daher durch $\frac{\alpha}{180}$ zu ersetzen.

15.4.2 \parametricplot

15.4.2.1 Lissajous Figur

Die aus der Physik oder Elektrotechnik bekannten Lissajoufiguren sind ein typischer Anwendungsfall für Gleichungen in Parameterform. Die hier angegebene Darstellung beruht auf den Funktionen:

Funktion	PostScript
$x = \sin 1.5t$	t 1.5 mul sin
$y = \sin\left(2t + \frac{\pi}{3}\right)$	t 2 mul 60 add sin



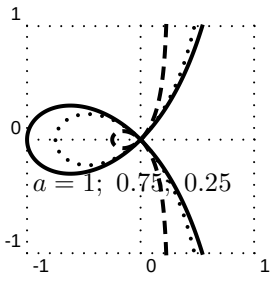

```
\psset{xunit=1.5cm,yunit=1.5cm}
\begin{pspicture}(-1.1,-1.1)(1.1,1.1)
  \psgrid[subgriddiv=0,griddots=10,%
    gridlabels=7pt](-1,-1)(1,1)
  \parametricplot[plotstyle=curve,%
    linewidth=1.5pt,plotpoints=200]{-360}{360}%
    {t 1.5 mul sin t 2 mul 60 add sin}
\end{pspicture}
```

Aufgrund der „Länge“ des Graphen wurde der Wert für `plotpoints` auf 200 gesetzt, sodass auch die „Ecken“ des Graphen mit starker Krümmung kontinuierlich erscheinen.

15.4.2.2 Strophoide

In den folgenden Gleichungen muss a durch einen Zahlenwert ersetzt werden:

Funktion	PostScript
$x(t) = \frac{a\left(t^2-1\right)}{t^2+1}$	t t mul 1 sub a mul t t mul 1 add div
$y(t) = \frac{at\left(t^2-1\right)}{t^2+1}$	t t mul 1 sub t mul a mul t t mul 1 add div




```
\psset{unit=1.5,subgriddiv=0,griddots=10,gridlabels=7pt}
\begin{pspicture}(-1.25,-1.25)(1.25,1.25)
  \psgrid(-1,-1)(1,1)
  \psset{plotpoints=200,linewidth=1.5pt}
  \def\PSfunction{t t mul 1 sub \param\space mul %
      t t mul 1 add div t t mul 1 sub t mul %
      \param\space mul t t mul 1 add div}
  \def\param{1}\parametricplot{-1.85}{1.85}{\PSfunction}
  \def\param{0.25}\parametricplot[linestyle=dashed]%
       {-4.5}{4.5}{\PSfunction}
  \def\param{0.75}\parametricplot[linestyle=dotted]%
       {-2.1}{2.1}{\PSfunction}
  \rput[l](-0.95,-0.4){$a=1;\ 0.75;\ 0.25$}
\end{pspicture}
```

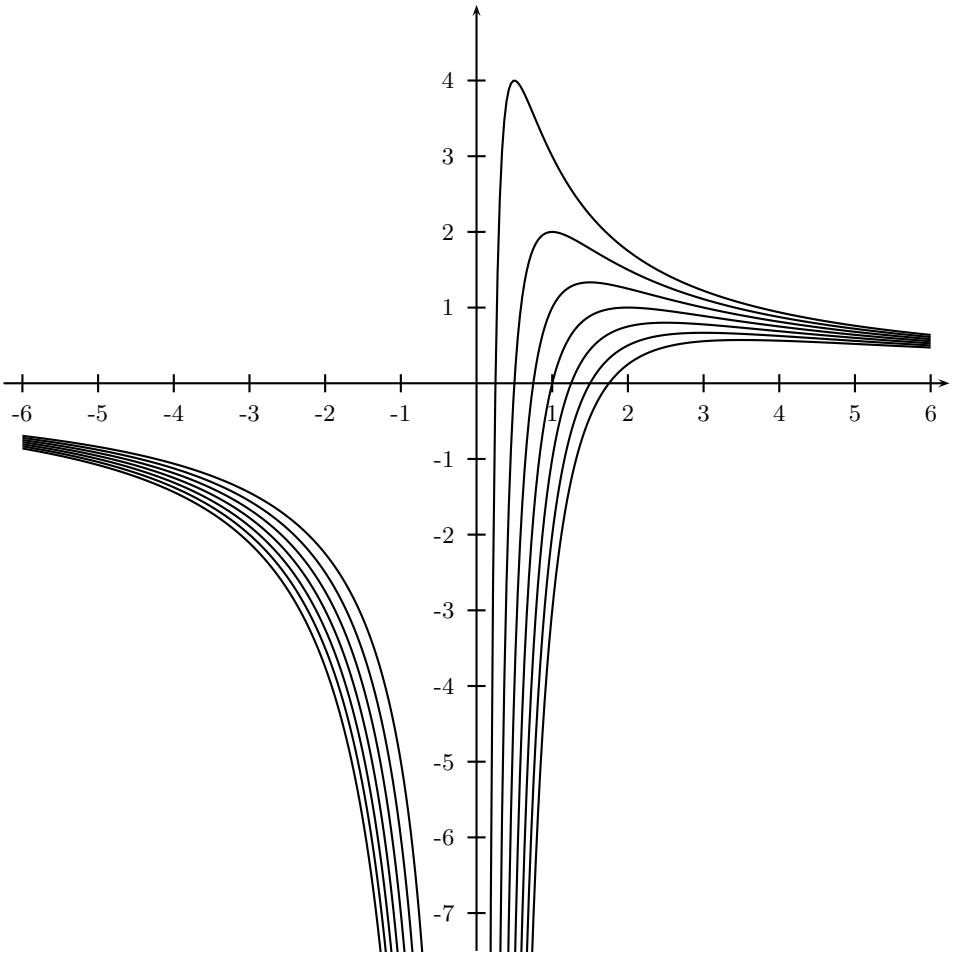


Abbildung 15.2: `bsp106.txt`: Die Kurvenschar der Gleichung $y = \frac{4x - t}{x^2}$ für $t \in \{1, ..., 7\}$.