

Grafiken in L^AT_EX mit PSTricks

Thorsten Vitt*

22. Oktober 2002

Dieser Text soll eine kurze Einführung in die Verwendung des Grafik-Pakets PSTricks für L^AT_EX geben. Es wird nur ein Ausschnitt aus der Funktionsvielfalt des Paketes dargestellt, vorwiegend mit Funktionen, die mir in Mathe und Theoretischer Informatik als nützlich erschienen.

Inhaltsverzeichnis

0	Prolog	2
0.1	Ansatz	2
0.2	Bezug und Dokumentation	2
0.3	Installation und Einbindung	2
1	Grundlegendes	2
1.1	Kommandos	3
1.2	Optionen	3
1.3	Koordinaten	3
2	Nodes	4
2.1	Beispiel: Endlicher Automat	4
3	Plotten von Funktionen	5
3.1	\psplot	5
3.2	PostScript-Funktionen	6
3.3	Achsen	6
3.4	Beispiel zum Funktionsplotten	7
4	Flächen beliebiger Form – Integrale usw.	7
4.1	\gsave und \grestore	8

*Rückfragen und Verbesserungsvorschläge bitte an vitt@informatik.hu-berlin.de

0 Prolog

0.1 Ansatz

PSTricks ist ein ziemlich leistungsfähiges Makro-Paket für \LaTeX zum Erstellen von Grafiken. Der Ansatz von PSTricks ist es, PostScript-Anweisungen mittels \LaTeX -`\specials` in das DVI-File zu packen. Es ist deshalb ein Treiber für das DVI-Konvertierungsprogramm nötig. Sowas gibts für dvips, nicht aber für pdfTeX oder dvipdfm, sodass man auf dvips angewiesen ist.

0.2 Bezug und Dokumentation

PSTricks gibts auf CTAN unter `graphics/pstricks`. Es gibt eine Homepage zu dem Paket bei der TUG: <http://www.tug.org/applications/PSTricks>.

Dateien mit Dokumentation dazu gibts ebenfalls auf CTAN bzw. der Homepage, z.B. `graphics/pstricks/obsolete/doc/*[2]`. Besonders die Datei `pst-quick.ps` sollte man sich ausgedruckt neben den Rechner legen (sind nur 8 A4-Seiten), da die eine Aufstellung der meisten PSTricks-Kommandos enthält. Leider ist die Doku unter `.../obsolete/` etwas veraltet.

Ausserdem kann ich das Kapitel zu PSTricks im \LaTeX Graphics Companion [1] empfehlen.

0.3 Installation und Einbindung

Die Installation ist detailliert in der README-Datei der PSTricks-Distribution beschrieben, es sind die Inhalte der Verzeichnisse `generic`, `latex` und `dvips` in passende Verzeichnisse zu kopieren. Bei mir landete der Inhalt von `generic` in `localtexmf/tex/generic/pstricks`, der von `latex` in `localtexmf/tex/latex/pstricks` und der von `dvips` in `localtexmf/dvips/pstricks`, danach war ein Update der Dateinamensdatenbank nötig (bei MiKTeX: `initexmf -u`).

Zur Benutzung ist das Paket `pstricks` und zudem ggf. noch das ein oder andere `pst-...`-Paket einzubinden. Wer z. B. Nodes und die Plotting-Funktionalitäten benutzen möchte, schreibt

```
\usepackage{pstricks, pst-node, pst-plot}
```

in die Präambel seines Dokuments.

1 Grundlegendes

Die meisten PSTricks-Kommandos benutzen *Koordinaten*. Grundlage ist ein kartesisches Koordinatensystem mit $(0,0)$ am aktuellen Arbeitspunkt von \TeX .

Meistens verwendet man PSTricks für (abgesetzte) Zeichnungen. Für die gibt es die Umgebung `pspicture`. In einer einfachen Form z.B.:

```
\begin{pspicture}(-1,-1)(3,4)
...
```


```
\end{pspicture}
```

Der Befehl erzeugt eine Box (d.h. reserviert Platz) mit einer Breite von 4 und einer Höhe von 5¹ und dem Nullpunkt jeweils eine Einheit von der linken unteren Ecke der Box nach links und nach rechts.

1.1 Kommandos

PSTricks-Befehle haben üblicherweise die Form

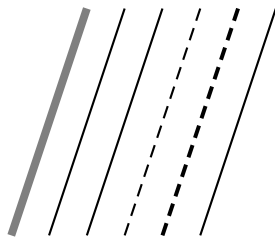
```
\kommando* [Optionen] {Pfeile bzw. Parameter} (Koordinaten) ...2
```

Dabei bedeutet der Stern üblicherweise, dass das Grafikobjekt gefüllt (und zwar mit dem Füllstil `solid`) statt durchsichtig gezeichnet werden soll, bzw. bei `pspicture` etc. dass am Rand „abgeschnitten“ werden soll. Parameter sind z.B. der irgendwo auszugebende Text bei den diversen `?put`-Kommandos, z.B. `\rput(0,0){Nullpunkt}`. Befehlen, die Linien o.ä. zeichnen, kann man mitgeben, wie die Pfeile aussehen sollen, z.B. erzeugt `\psline{|->>}(0,0)(2em,1ex)` den Pfeil .

1.2 Optionen

Es gibt viele, viele Optionen, mit denen man PSTricks beeinflussen kann. Optionen sind immer `key=value`-Paare, und es gibt für alle Optionen Voreinstellungen. Optionen kann man entweder einem Befehl als optionales Argument mitgeben oder über den `\psset`-Befehl angeben. In beiden Fällen werden mehrere Optionen durch Kommata getrennt.

Optionen in eckigen Klammern gelten nur für den jeweiligen Befehl. `\psset` wirkt bis zum Ende der aktuellen Gruppe (`{ }`) bzw. des aktuellen Environments. Beispiel:



```
\begin{pspicture}(0,0)(5,3)
  \psline[linewidth=3pt,linecolor=gray](0,0)(1,3)
  \psline(1,0)(1,3)
  {
    \psline(2,0)(2,3)
    \psset{linestyle=dashed}
    \psline(3,0)(3,3)
    \psline(3.5,0)(4.5,3)
  }
  \psline(4,0)(5,3)
\end{pspicture}
```

1.3 Koordinaten

Koordinaten werden üblicherweise in der Form (x, y) angegeben, wobei x und y kartesische Koordinaten bezeichnen. Entweder steht bei einer Koordinate eine Einheit (z.B.

¹ 5 was? Normalerweise Zentimeter. Siehe dazu aber 1.3

² optionale Argumente unterstrichen

1ex oder 0.2cm), oder es wird – und das ist der übliche Weg – die aktuelle Koordinateneinheit gewählt. Die wird über die Optionen xunit, yunit und runit für x -, y - bzw. sonstige Koordinaten gesetzt, oder über unit für alle drei gemeinsam. So können Grafiken skaliert oder gestaucht bzw. gedehnt werden.

2 Nodes

PSTricks bietet ein weiteres mächtiges Feature, das bei bestimmten Zeichnungen viel Handarbeit ersparen kann: *Nodes*.

Unter einem Node versteht man, allgemein gesprochen, ein mit einem Namen versehenes Zeichnungsobjekt. Diese Nodes können dann mit Verbindungslinien verschiedener Gestalt verbunden werden, und sowohl die Linien als auch die Nodes selbst können beschriftet werden. Die genaue Positionierung bzw. Gestaltung der Linien und Beschriftungen wird dabei von PSTricks berechnet. Das lässt sich natürlich durch Parameter beeinflussen, man muss sich halt nur keine Gedanken machen, wo genau die Linien bzw. Labels nun hin müssen.

Es gibt eine Reihe von Befehlen, die unterschiedliche Nodes erzeugen können.

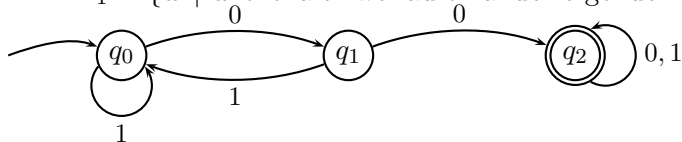
— TODO —

2.1 Beispiel: Endlicher Automat

Das folgende Beispiel verdeutlicht vielleicht ein wenig die Anwendung von Nodes. Wir setzen zunächst mittels `\cnodeput` die Zustände ins Koordinatensystem. `\cnodeput` platziert Text an die angegebenen Koordinaten, umgibt den Text mit einem Kreis und benennt das ganze dann als Node.

Dann werden mit `\ncarc` die Pfeile für δ gemalt.

$L_1 = \{w \mid w \text{ enthält zwei aufeinanderfolgende Nullen}\}$:



% Ein paar Grundeinstellungen:

```
\psset{arrows=->,      % -> als Pfeil-Grundeinstellung
       unit=3cm,        % Standardgröße für die Einheit
       arcangle=20,     % Wie krumm sind die \ncarcs (in Grad)
       labelsep=2.5pt,  % Labels sind 2.5pt vom zu belabelnden entfernt
       subgriddiv=10,   % für \psgrid
       shortput=nab}    % ^ und _ für \naput und \nbput
```

```
$L_1 = \{w \mid w \text{ \texttt{enth\"alt zwei aufeinanderfolgende Nullen}}\}:
```

```
\begin{pspicture}(-0.2,-0.4)(2.5,0.2) % Begrenzungen des Images
```

```
    %\psgrid % Auskommentieren zum Austesten der Dimensionen
```

```

\pnode(-0.5, 0){0}      % Leerer Node für Pfeil zum Anfangszustand

% Malen wir die Nodes:
\cnodeput(0,0){q0}{\$q_0\$}
\cnodeput(1,0){q1}{\$q_1\$}
\cnodeput[doubleline=true](2,0){q2}{\$q_2\$}

{ \small      % Beschriftungen der Verbindungen sollen kleiner gesetzt werden

% Nun die Verbindungen, gleich mit Beschriftungen:
\ncarc{0}{q0}      % → Anfangszustand

\ncarc{q0}{q1}^{\$0\$}
\ncarc{q1}{q2}^{\$0\$}
\ncarc{q1}{q0}^{\$1\$}

% die „Loops“ – q0-q0-Verbindungen etc.
\ncircle[angle=180]{q0}{0.4cm}_{\$1\$}      % Der soll nach unten gehen
\ncircle[angle=-90]{q2}{0.4cm}_{\$0,1\$}      % und der rechts des Zustands
}
\end{pspicture}

```

3 Plotten von Funktionen

Ein Funktionsplotter ist auch schon drin, nämlich im Paket `pst-plot`.

Es gibt im Grunde zwei Möglichkeiten, eine Funktion von PSTricks plotten zu lassen:

- Man generiert eine Wertetabelle, zum Beispiel mit *gnuplot*, und lässt die dann mit `\fileplot` etc. zeichnen.
- Man benutzt `\psplot` und gibt die Funktion direkt als PostScript-Code an.

Auf die erste Methode will ich nicht weiter eingehen, da ich das fürs Plotten einfacher mathematische Funktionen für zu unflexibel und aufwändig halte.

3.1 `\psplot`

```
\psplot*[Optionen]{ $x_{min}$ }{ $x_{max}$ }{Funktion  $f(x)$ }
```

Dieser Befehl zeichnet eine Funktion $f(x)$, die als Postscriptcode im dritten Argument gegeben ist, im aktuellen Koordinatensystem, und zwar im Intervall $[x_{min}, x_{max}]$. Dabei sind ein paar Dinge zu beachten:

- Der als Funktion angegebene Code wird direkt als PostScript-Code ausgeführt. Das bedeutet auch, dass Fehler hier nicht von \TeX , sondern erst vom PostScript-Interpreter gemeldet werden. Man sollte sich also immer mittels Ghostscript oder so vergewissern, dass sein Dokument auch funktioniert.

Der folgende Code wäre z.B. problematisch, obschon \TeX keinen Fehler meldet³:

```
\begin{pspicture}(-1,0)(1,1)
  \psplot{-1}{1}{1 x div}
\end{pspicture}
```

- Bei Funktionen, die innerhalb des gezeichneten Intervalls gegen unendlich gehen, sollte man das Intervall entsprechend anpassen oder `pspicture*` verwenden, damit nicht wild Linien übers ganze Blatt gemalt werden.
- Es wird ausschließlich die Funktion gezeichnet. Sowohl das Malen der Koordinatenachsen als auch Beschriftungen müssen extra vorgenommen werden.
- Es wird immer das aktuelle Koordinatensystem verwendet. Das kann man natürlich verändern, z.B. im optionalen Argument von `psplot` mit Optionen wie `units` oder `origin`.

— TODO: Beispiel mit Bernstein-Polynomen ergänzen —

3.2 PostScript-Funktionen

PostScript ist eine Postfix-Sprache, und der Teilbereich den wir brauchen ist ziemlich einfach. Die Eingabe wird von links nach rechts gelesen, kommt eine Zahl, so wird die in den Keller geschubst (x als Laufvariable betrachten wir hier als Zahl), kommt ein Operator, so holt dieser ein oder mehrere Zeichen von oben aus dem Keller, rechnet und schreibt das Ergebnis wieder drauf. Nach Abarbeitung des Funktionsarguments von `\psplot` darf noch genau eine Zahl im Keller stehen, das ist dann die passende y -Koordinate zum aktuellen x .

Operatoren sind kurze Kleinbuchstabenfolgen, die wichtigsten hier sind wohl `add`, `sub`, `mul`, `div`, `exp` ($x_1\ x_2\ \text{add} \equiv x_1 + x_2$, die übrigen stehen analog für $x_1 - x_2$, $x_1 \cdot x_2$, $\frac{x_1}{x_2}$, $x_1^{x_2}$). `sin` und `cos` operieren auf Grad (und nehmen nur ein Argument).

Eine komplette Übersicht über die Operatoren findet man im PostScript-Referenzhandbuch — TODO: Details ergänzen —.

3.3 Achsen

Für Koordinatenachsen gibts den Befehl `\psaxes`:

```
\psaxes*{Pfeile}(x_0,y_0)(x_1,y_1)(x_2,y_2)
```

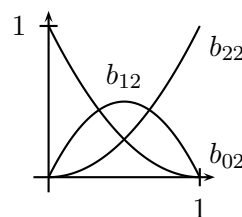
Zeichnet Koordinatenachsen mit einer Ausdehnung von (x_1, y_1) bis x_2, y_2 und einem Schnittpunkt bei (x_0, y_0) . (x_0, y_0) sind standardmäßig (x_1, y_1) , und diese sind standardmäßig $(0, 0)$.

³ $1\ x\ \text{div} = \frac{1}{x}$: $x \in [-1, 1]$ wird bei $x = 0$ problematisch...

3.4 Beispiel zum Funktionsplotten

Die nebenstehende Grafik (die Bernstein-Polynome für $n = 2$) lässt sich mit Hilfe von `\psplot` mit nur wenigen Zeilen erledigen. Die Hauptarbeit besteht wohl darin, die Funktionen auszurechnen und nach PostScript umzuwandeln ;-)

Beachte auch die Verwendung von `\uput` zum Platzieren der Beschriftungen.



```
{ % Wir beschränken die Einstellungen auf diese Grafik
  \psset{unit=2cm}           % Koordinateneinheit. Wir zeichnen ja nur auf [0,1]

  % Wir erzeugen das Bild. Es soll groß genug sein, um die
  % Achsenbeschriftungen ebenfalls aufzunehmen
  \begin{pspicture}(-0.4,-0.4)(1.2,1.2)

    % Nun müssen wir Koordinatenachsen malen. Das Beschriften und
    % Tick-Zeichnen darf dabei ruhig automatisch geschehen.
    \psaxes[ticks=all,labels=all]{->}(0,0)(-0.1,-0.1)(1.1,1.1)

    % Nun werden die drei Funktionen gemalt.
    \psplot{0}{1}{1 x sub 2 exp} % (1-x)^2 im Intervall [0,1]
    \uput[ur](1,0){$b_{0\ 2}$} % b02 soll oben rechts (up right) von (1,0) erscheinen
    % Analog -x^2 * 2 + 2 * x, Beschriftung unten rechts
    \psplot{0}{1}{x 2 exp 2 mul neg 2 x mul add} \uput[dr](1,1){$b_{2\ 2}$}
    \psplot{0}{1}{x 2 exp} \uput[u](0.5,0.5){$b_{1\ 2}$}
  \end{pspicture}
}
```

4 Flächen beliebiger Form – Integrale usw.

Eigentlich ist die Überschrift nicht ganz korrekt, denn es geht hier um das Erstellen eigener Grafikobjekte (die natürlich eine beliebige Form haben und gefüllt werden können). Ich benutze das eigentlich nur für Integrale usw., also für Flächen einer Form, die vorwiegend durch `\psplot` bestimmt wird.

PSTricks bietet dafür den Befehl `\pscustom` an:

```
\pscustom*[Optionen]{Befehle}
```

Befehle sind PSTricks-Zeichenbefehle (und ein paar zusätzliche).

Die Möglichkeiten mit `\pscustom` sind sehr umfangreich. Ich werde hier nur das Grundkonzept darlegen, Details findet man in [2].

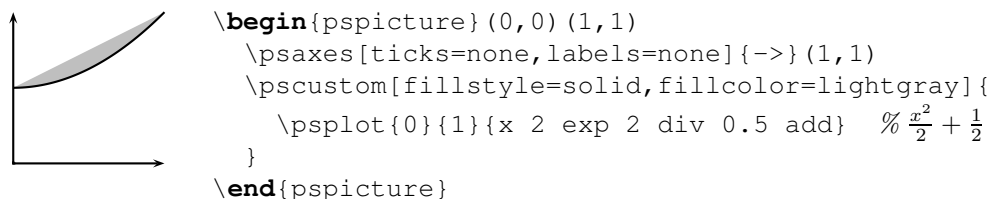
Die Zeichenbefehle verhalten sich innerhalb eines `\pscustom`-Kommandos anders als sonst: Sie zeichnen nicht direkt, sondern sie formen einen *Pfad*. Dabei sollte man nur die

Befehle verwenden, die *offene Kurven* zeichnen – also Linien, Kurven, Bögen, Funktionsgraphen⁴. Der jeweils letzte Punkt so einer Figur ist der *aktuelle Punkt* (current point), der normalerweise entweder den Anfangspunkt des nächsten Objektes im `\pscustom` darstellt oder mit diesem verbunden wird.

Der Befehl `\stroke` zeichnet den aktuellen Pfad nach (und zwar mit den aktuellen Linieneinstellungen, die durch das optionale Argument des Befehles geändert werden können), der Befehl `\fill` füllt die Fläche analog dazu. Beide Befehle werden am Ende des `\pscustom`-Befehls automatisch aufgerufen.

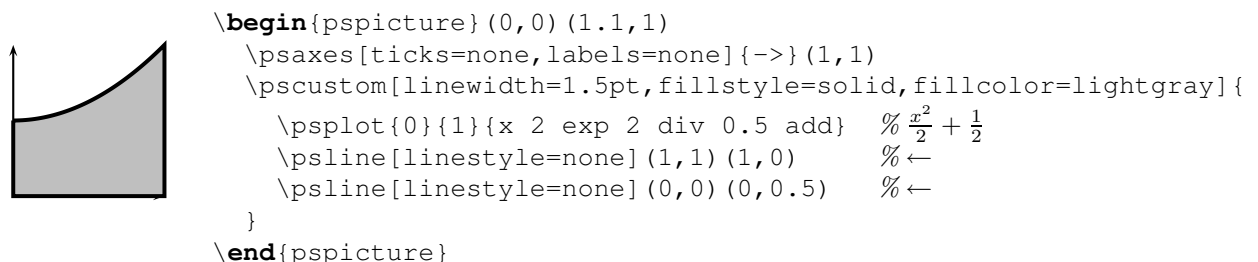
Beispiel

Wir wollen $\int_0^1 \frac{x^2}{2} + \frac{1}{2} dx$ darstellen. Der erste naive Ansatz:



Es wird nicht die gewünschte Fläche gefüllt. Vielmehr wird der Endpunkt der Zeichnung (nämlich $(1,1)$), d.h. der letzte aktuelle Punkt, mit dem Anfangspunkt verbunden und die entstandene Fläche gefüllt.

Die Lösung dazu besteht darin, die gesuchte Fläche tatsächlich zu umzeichnen, indem wir an den Intervallgrenzen Funktion und Koordinatenachse mit einer – idealerweise unsichtbaren – Linie verbinden:



So ganz das gewünschte Resultat haben wir hier noch nicht erreicht: Die Intervallgrenzen sind noch sichtbar. Klar, denn in den mit `←` gekennzeichneten Zeilen haben wir den falschen Ansatz gewählt, um unsichtbare Linien zu zeichnen: Der `\psline`-Befehl verändert innerhalb von `\pscustom` schließlich nur den Pfad, zeichnet nicht selber.

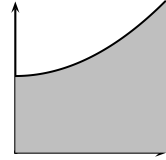
4.1 `\gsave` und `\grestore`

Die Befehle `\gsave` und `\grestore` bringen die Lösung unseres Problems. `\gsave` speichert den aktuellen Zustand des `\pscustom`-Objekts, also Pfad, Koordinatensystem, aktuellen Punkt und so weiter (auf einem Stack), `\grestore` stellt das alles wieder

⁴mit `plotstyle=line` (Voreinstellung) oder `curve`

her. `\gsave` und `\grestore` müssen *immer paarweise* und korrekt geschachtelt, unter Berücksichtigung von T_EX-Gruppen auftreten!

```
\begin{pspicture}(0,0)(1.1,1)
  \psaxes[ticks=none,labels=none]{->}(1,1)
  \pscustom[linewidth=1.5pt,fillstyle=solid,fillcolor=lightgray]{
    \gsave      % Noch ist nichts geschehen → speichern
    \psplot{0}{1}{x 2 exp 2 div 0.5 add} %  $\frac{x^2}{2} + \frac{1}{2}$ 
    \stroke     % Die Kurve soll gezeichnet werden
    \psline(1,1)(1,0)
    \psline(0,0)(0,0.5)
    \fill      % Die komplette Fläche füllen
  }
  \grestore % und den „leeren“ Zustand von oben wiederherstellen
\end{pspicture}
```



Man beachte im letzten Beispiel noch, dass die Füllung sowohl die Hälfte der Linie des Funktionsgraphen als auch der Koordinatenachsen übermalt. Das liegt einfach an der Reihenfolge, in der das im Quelltext steht (und dann auch gezeichnet wird). Zur Lösung dieses Problems kann man die Reihenfolge der einzelnen Zeichenbefehle entsprechend anpassen, es spricht auch nichts dagegen, auf das `\stroke` in Zeile 6 zu verzichten und den `\psplot`-Befehl nach dem `\pscustom` zu wiederholen.

Im Allgemeinen würde ich jedoch empfehlen, auf den Füllstil `solid` ganz zu verzichten und lieber `hlines` etc. zu verwenden, da das Kopierer-freundlicher ist.

Literatur

- [1] Michael Goossens, Sebastian Rahtz, and Frank Mittelbach. *The L^AT_EX Graphics Companion*. Addison Wesley Longman, Inc., 1997.
- [2] Timothy Van Zandt. PSTricks: PostScript macros for generic T_EX. User's guide. Teil der PSTricks-Distribution. CTAN z.B.: <ftp://ftp.dante.de/tex-archive/graphics/pstricks/obsolete/doc/>, March 1993.