PSTricks

pst-intersect
Berechnen der Schnittpunkte beliebiger Kurven

v0.3

2014/03/04

Paketautor:
Christoph Bersch

Inhaltsverzeichnis

1. Einführung

1.1. Über das Paket

`pst-intersect` ist ein PSTricks-Paket zur Berechnung der Schnittpunkte von Bézier-Kurven und beliebigen Postscript-Pfaden.

Beachten Sie, dass die Paket-Versionen 0.x sich in einem experimentellen Status befinden, und sich grundlegende Änderungen ergeben können, die zur Vorgängerversion inkompatibel sind.

1.2. Anforderungen

`pst-intersect` benötigt aktuelle Versionen der Pakete `pstricks`, `pst-node` und `pst-func`.

Alle PSTricks-Pakete machen regen Gebrauch von der Postscript-Sprache, so dass der typische Arbeitsfluss `latex`, `dvips` und ggf. `ps2pdf` umfasst. Es gibt viele alternative Methoden um die Dokumente zu kompilieren.[1]

1.3. Verbreitung und Installation

Dieses Paket ist auf CTAN[2] erhältlich.

Das `pst-intersect`-Paket umfasst die zwei Hauptdateien `pst-intersect.ins` und `pst-intersect.dtx`. Durch Aufrufen von `tex pst-intersect.ins` werden die drei folgenden Dateien erzeugt:

- `pst-intersect.pro`: die Postscript Prologdatei
- `pst-intersect.sty`: die LaTeX-Stildatei

[1] http://tug.org/PSTricks/main.cgi?file=pdf/pdfoutput
[2] http://mirror.ctan.org/help/Catalogue/entries/pst-intersect.html

- `pst-intersect.tex`: die TEX-Datei

Speichern Sie diese Dateien in einem Verzeichnis der Teil Ihres lokalen TEX-Baums ist.

Vergessen Sie nicht `texhash` aufzurufen um den Baum zu aktualisieren. MiKTEX-Benutzer müssen die Dateinamen-Datenbank (FNDB) aktualisieren.

Detailliertere Information finden Sie in der Dokumentation Ihrer LATEX-Distribution über die Installation in den lokalen TEX-Baum.

1.4. Lizenz

1.5. Danksagung

Ich danke Marco Cecchetti, dessen `lib2geom`-Bibliothek[4] mir als Vorlage für einen Großteil des Postscript-Kodes für den Bézier-Clipping-Algorithmus diente. Außerdem gilt mein Dank William A. Casselman, für seine Erlaubnis, den Quicksort-Kode und den Kode zur Berechung der konvexen Hüllen aus seinem Buch „Mathematical Illustration“ verwenden zu dürfen[5]. Der Dokumentationsstil ist eine Mischung aus der `pst-doc` Klasse (Herbert Voß) und dem `ltxdockit` Paket für die `biblatex` Dokumentation (Philipp Lehmann).

[3] http://www.latex-project.org/lppl.txt
[4] http://lib2geom.sourceforge.net/
[5] http://www.math.ubc.ca/~cass/graphics/text/www/

2. Benutzung

Das `pst-intersect`-Paket kann Schnittpunkte von beliebigen Postscript-Pfaden berechnen. Diese setzen sich nur aus drei primitiven Operation zusammen: Linien (`lineto`), Bézier-Kurven dritter Ordnung (`curveto`) und Sprüngen (`moveto`). Speziellere Konstruktionen, wie z.B. Kreise werden intern zu `curveto`-Anweisungen umgewandelt. Über diese Kommandos hinaus kann `pst-intersect` auch Bézier-Kurven bis neunter Ordnung verwenden. Das diese keine primitiven Postscript-Pfadelemente darstellen, müssen sie gesondert behandelt werden.

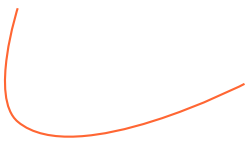
Der allgemeine Arbeitsablauf besteht darin eine oder mehrere Kurven oder Pfade zu speichern, und danach die Schnittpunkte zu berechnen. Anschließend können die Schnittpunkte als normale PSTricks-Knoten verwendet werden, oder Abschnitte der Kurven und Pfade nachgezogen werden (z.B. zwischen zwei Schnittpunkten).

2.1. Speichern von Pfaden und Kurven

`\pssavepath`[⟨*options*⟩]{⟨*curvename*⟩}{⟨*commands*⟩}

Speichert den gesamten Pfad, der durch ⟨*commands*⟩ erstellt wird, unter Verwendung des Namens ⟨*curvename*⟩. Das Makro funktioniert genauso wie `\pscustom`, und kann daher auch nur die darin erlaubten Kommandos verarbeiten.

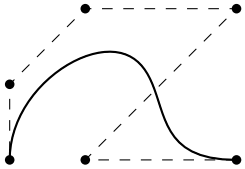
In den Standardeinstellungen wird der entsprechende Pfad auch gleich gezeichnet, was mit ⟨*options*⟩ beeinflusst werden kann. Mit `linestyle=none` wird das unterbunden.



```
\begin{pspicture}(3,2)
  \pssavepath[linecolor=DOrange]{MyPath}{%
    \pscurve(0,2)(0,0.5)(3,1)
  }%
\end{pspicture}
```

\pssavebezier[⟨*options*⟩]{⟨*curvename*⟩}(⟨X_0⟩)...(⟨X_n⟩)

Die Postscript-Sprache unterstützt nur Bézier-Kurven dritter Ordnung. Mit dem Makro \pssavebezier können Bézier-Kurven bis neunter Ordnung definiert werden. Die angegebenen Knoten sind die Kontrollpunkte der Kurve, für eine Kurve n-ter Ordnung werden $(n + 1)$ Kontrollpunkte benötigt. Die Darstellung der Kurve erfolgt mit dem Makro \psBezier aus dem pst-func-Paket.

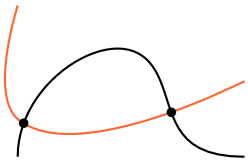


```
\begin{pspicture}(3,2)
  \pssavebezier[showpoints]{MyBez}(0,0)(0,1)(1,2)(3,2)(1,0)(3,0)
\end{pspicture}
```

2.2. Schnittpunkte berechnen

\psintersect{⟨*curveA*⟩}{⟨*curveB*⟩}

Nachdem Sie nun Pfade oder Kurven gespeichert haben, können Sie deren Schnittpunkte berechnen. Das geschieht mit dem Makro \psintersect. Dieses benötigt als Argumente zwei Namen von Pfaden oder Kurven (Das Argument ⟨*curvename*⟩ der beiden \pssave∗ Makros).



```
\begin{pspicture}(3,2)
  \pssavepath[linecolor=DOrange]{MyPath}{\pscurve(0,2)(0,0.5)(3,1)}
  \pssavebezier{MyBez}(0,0)(0,1)(1,2)(3,2)(1,0)(3,0)
  \psintersect[showpoints]{MyPath}{MyBez}
\end{pspicture}
```

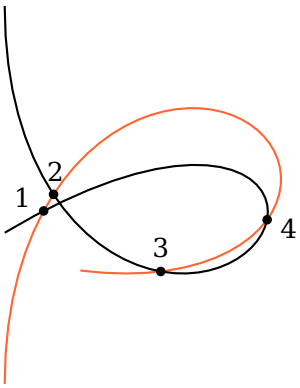
Der PSTricks-Parameter showpoints steuert dabei, ob die Schnittpunkte angezeigt werden.

name=⟨*string*⟩ default: @tmp

Die berechneten Schnittpunkte können unter einem hier angegebenen Namen gespeichert und zu einem späteren Zeitpunkt verwendet werden (siehe Kap. 2.4).

saveintersections=true, false default: true

Ist dieser Schalter gesetzt, dann werden die Schnittpunkte als PSTricks-Knoten unter den Namen ⟨*name*⟩1, ⟨*name*⟩2 ... gespeichert. Die Nummerierung erfolgt aufsteigend nach dem Wert der x-Koordinate.

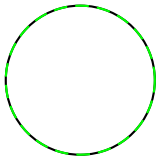



```
\begin{pspicture}(5,5)
  \pssavebezier[linecolor=DOrange]{A}%
           (0,0)(0,5)(5,5)(5,1)(1,1.5)
  \pssavebezier{B}(0,5)(0,0)(5,0)(5,5)(0,2)
  \psintersect[name=C, showpoints]{A}{B}
  \uput[150](C1){1}
  \uput[85](C2){2}
  \uput[90](C3){3}
  \uput[-20](C4){4}
\end{pspicture}
```

2.3. Darstellung gespeicherter Pfade

`\pstracecurve`[⟨*options*⟩]{⟨*curvename*⟩}

Gespeicherte Pfade und Kurven können mit diesem Makro nachträglich gezeichnet werden.

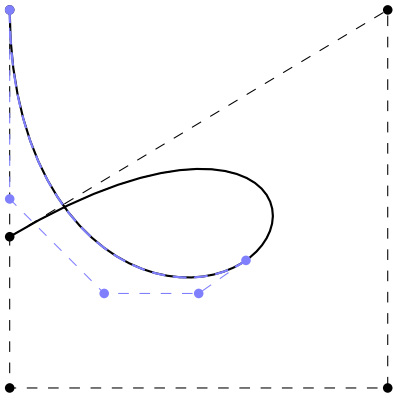


```
\begin{pspicture}(2,2)
  \pssavepath{Circle}{\pscircle(1,1){1}}
  \pstracecurve[linestyle=dashed, linecolor=green]{Circle}
\end{pspicture}
```

`tstart`=⟨*num*⟩

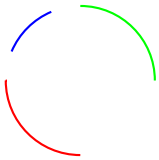
`tstop`=⟨*num*⟩

Unter Verwendung dieser beiden Parameter können auch Abschnitte von Pfaden und Kurven gezeichnet werden. Bei Bézier-Kurven ist der Parameterbereich $[0, 1]$, wobei 0 dem Anfang der Kurve, also dem ersten bei `\pssavebezier` angegebenen Knoten entspricht.



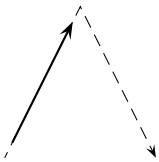
```
\begin{pspicture}(5,5)
  \psset{showpoints}
  \pssavebezier{B}(0,5)(0,0)(5,0)(5,5)(0,2)
  \pstracecurve[linestyle=dashed, linecolor=blue!50,
              tstart=0, tstop=0.5]{B}
\end{pspicture}
```

Pfaden können aus mehr als einem Abschnitt bestehen, der Bereich ist also $[0, n]$, wobei n die Anzahl der Pfadabschnitte ist. Dabei ist zu beachten, dass z.B. `\pscurve`-Pfade oder auch Kreise und Kreisbögen aus mehreren Abschnitten bestehen.

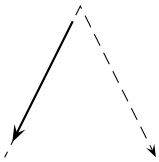


```
\begin{pspicture}(2,2)
  \pssavepath[linestyle=none]{Circle}{\pscircle(1,1){1}}
  \pstracecurve[tstart=0, tstop=1, linecolor=green]{Circle}
  \pstracecurve[tstart=2, tstop=3, linecolor=red]{Circle}
  \pstracecurve[tstart=1.25, tstop=1.75, linecolor=blue]{Circle}
\end{pspicture}
```

Beachten Sie, dass die Reihenfolge von `tstart` und `tstop` eine Rolle spielt. Ist `tstart` > `tstop` dann wird die Pfadrichtung umgekehrt.



```
\begin{pspicture}(2,2)
  \psset{arrows=->, arrowscale=1.5}
  \pssavepath[linestyle=dashed,
      linewidth=0.5\pslinewidth]{A}{\psline(0,0)(1,2)(2,0)}
  \pstracecurve[tstart=0.1, tstop=0.9]{A}
\end{pspicture}
```



```
\begin{pspicture}(2,2)
  \psset{arrows=->, arrowscale=1.5}
  \pssavepath[linestyle=dashed,
      linewidth=0.5\pslinewidth]{A}{\psline(0,0)(1,2)(2,0)}
  \pstracecurve[tstart=0.9, tstop=0.1]{A}
\end{pspicture}
```

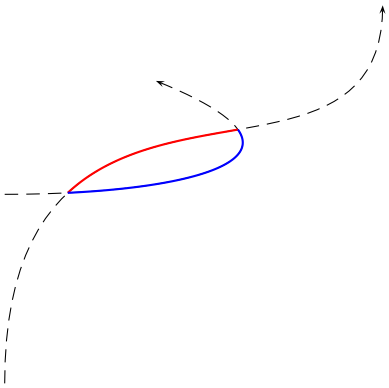
2.4. Darstellung gespeicherter Schnitte

\pstracecurve[⟨*options*⟩]{⟨*intersection*⟩}{⟨*curvename*⟩}

istart=⟨*num*⟩

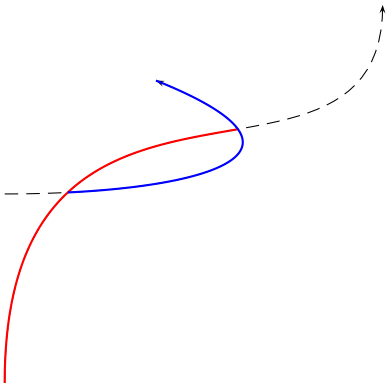
istop=⟨*num*⟩

Dieser beiden Parameter können auch verwendet werden um Abschnitte von Pfaden und Kurven zwischen Schnittpunkten zu zeichnen. Die Schnittpunkte werden dabei angefangen bei 1 in aufsteigender Reihenfolge entlang der Kurve durchnummeriert.



```
\begin{pspicture}(5.2,5.2)
\pssavebezier[linewidth=0.5\pslinewidth, ↵
    linestyle=dashed, arrows=->]{A}(0,0)(0,5)(5,2)(5,5)
\pssavebezier[linewidth=0.5\pslinewidth, ↵
    linestyle=dashed, ↵
    arrows=->]{B}(0,2.5)(2.5,2.5)(4.5, 3)(2,4)
\psintersect[linecolor=green!70!black, name=C]{A}{B}
\pstracecurve[linecolor=red, istart=1, istop=2]{C}{A}
\pstracecurve[linecolor=blue, istart=1, istop=2]{C}{B}
\end{pspicture}
```

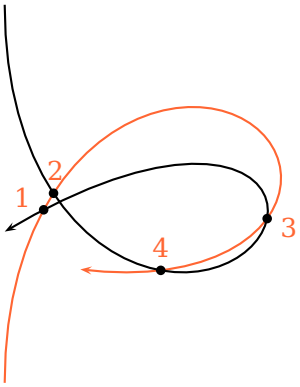
Wird nur ein Wert angegeben, beispielsweise istop, dann wird die Kurve vom Anfang bis zum entsprechenden Schnittpunkt gezeichnet. Wird nur istart angegeben, dann endet die Kurve am Ende. Die Parameter istart bzw. istop können mit tstart bzw. tstop kombiniert werden.



```
\begin{pspicture}(5.2,5.2)
\pssavebezier[linewidth=0.5\pslinewidth, ↵
    linestyle=dashed, arrows=->]{A}(0,0)(0,5)(5,2)(5,5)
\pssavebezier[linewidth=0.5\pslinewidth, ↵
    linestyle=dashed, ↵
    arrows=->]{B}(0,2.5)(2.5,2.5)(4.5, 3)(2,4)
\psintersect[linecolor=green!70!black, name=C]{A}{B}
\pstracecurve[linecolor=red, istop=2]{C}{A}
\pstracecurve[linecolor=blue, istart=1]{C}{B}
\end{pspicture}
```

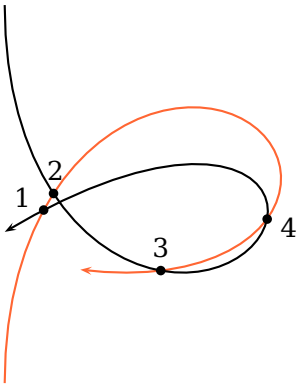
\psGetIsectCenter{⟨*intersection*⟩}{⟨*curvename*⟩}{⟨*n*⟩}

Lädt die Koordinaten des n-ten Schnittpunkts des Pfades ⟨*curvename*⟩ in der Schnittmenge ⟨*intersection*⟩. Dabei startet die Nummer der Schnittpunkte bei ⟨*n*⟩$= 1$ und wird in Pfadrichtung durchnummeriert.




```
\begin{pspicture}(4,5)
  \pssavebezier[linecolor=DOrange, arrows=->]{A}%
           (0,0)(0,5)(5,5)(5,1)(1,1.5)
  \pssavebezier[arrows=->]{B}(0,5)(0,0)(5,0)(5,5)(0,2)
  \psintersect[name=C, showpoints]{A}{B}
  \color{DOrange}
  \uput[150](!\psGetIsectCenter{C}{A}{1} I-C1.x I-C1.y){1}
  \uput[85](!\psGetIsectCenter{C}{A}{2} I-C2.x I-C2.y){2}
  \uput[-20](!\psGetIsectCenter{C}{A}{3} I-C3.x I-C3.y){3}
  \uput[90](!\psGetIsectCenter{C}{A}{4} I-C4.x I-C4.y){4}
\end{pspicture}
```

Wenn kein Kurvenname ⟨*curvename*⟩ angegeben wird, dann werden die Punkte nach aufsteigender x-Koordinate sortiert:

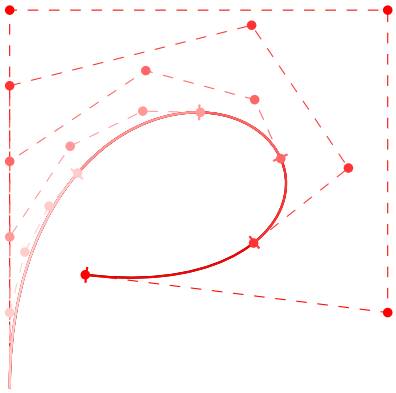



```
\begin{pspicture}(4,5)
  \pssavebezier[linecolor=DOrange, arrows=->]{A}%
           (0,0)(0,5)(5,5)(5,1)(1,1.5)
  \pssavebezier[arrows=->]{B}(0,5)(0,0)(5,0)(5,5)(0,2)
  \psintersect[name=C, showpoints]{A}{B}
  \uput[150](!\psGetIsectCenter{C}{}{1} I-C1.x I-C1.y){1}
  \uput[85](!\psGetIsectCenter{C}{}{2} I-C2.x I-C2.y){2}
  \uput[90](!\psGetIsectCenter{C}{}{3} I-C3.x I-C3.y){3}
  \uput[-20](!\psGetIsectCenter{C}{}{4} I-C4.x I-C4.y){4}
\end{pspicture}
```

Die Schnittpunkte können auch mit saveintersections und saveNodeCoors geladen werden. Diese werden dann ebenfalls nach aufsteigender x-Koordinate nummeriert.

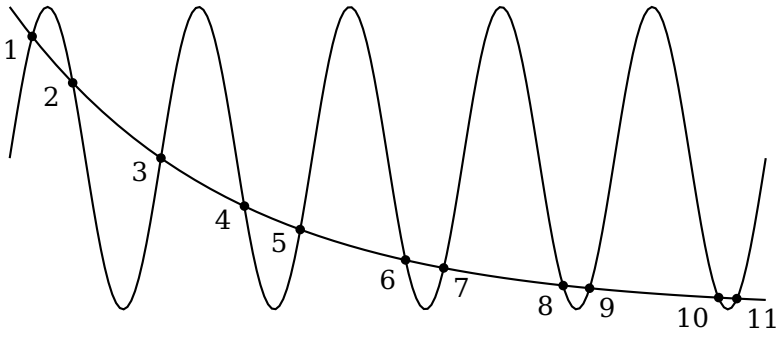
3. Beispiele

Bsp. 3.1



```
\begin{pspicture}(5,5)
  \pssavebezier{A}(0,0)(0,5)(5,5)(5,1)(1,1.5)
  \multido{\i=100+-20,\r=1+-0.2}{5}{%
    \pstracecurve[linecolor=red!\i, tstop=\r,
        arrows=-|, showpoints]{A}
  }%
\end{pspicture}
```

Bsp. 3.2: Mit diesem Paket können auch die Schnittpunkte von Funktionen berechnet werden, die mit `\psplot` gezeichnet werden. Dabei ist zu beachten, dass beide einzelnen Kurven aus `plotpoints` Abschnitten bestehen, von denen jeder mit jedem geschnitten wird, was zu langen Berechnungen führen kann.




```
\begin{pspicture}(10,4.4)
  \pssavepath{A}{\psplot[plotpoints=200]{0}{10}{x 180 mul sin 1 add 2 mul}}
  \pssavepath{B}{\psplot[plotpoints=50]{0}{10}{2 x neg 0.5 mul exp 4 mul}}
  \psintersect[name=C, showpoints]{A}{B}
  \multido{\i=1+1}{5}{\uput[210](C\i){\i}}
  \multido{\i=6+2,\ii=7+2}{3}{\uput[225](C\i){\i}\uput[-45](C\ii){\ii}}
\end{pspicture}
```

A. Versionsgeschichte

Diese Versionsgeschichte ist eine Liste von Änderungen, die für den Nutzer des Pakets von Bedeutung sind. Änderungen, die eher technischer Natur sind und für den Nutzer des Pakets nicht relevant sind und das Verhalten des Pakets nicht ändern, werden nicht aufgeführt. Wenn ein Eintrag der Versionsgeschichte ein Feature als *improved* oder *extended* bekannt gibt, so bedeutet dies, dass eine Modifikation die Syntax und das Verhalten des Pakets nicht beeinflusst, oder das es für ältere Versionen kompatibel ist. Einträge, die als *deprecated*, *modified*, *renamed*, oder *removed* deklariert sind, verlangen besondere Aufmerksamkeit. Diese bedeuten, dass eine Modifikation Änderungen in existierenden Dokumenten mit sich ziehen kann.

0.3 2014-03-04

Fixed `\psGetIsectNode` to work with more than one segment.
Modified `\psGetIsectNode` and the naming conventions of the variables.
Fixed missing support for `pst-node`'s `saveNodeCoors` parameter to `saveintersections`.

0.2 2014-02-26

Added support for `arrows` parameter to `\pstracecurve`.
Modified parameters `tstart`, `tstop`, `istart` and `istop` to respect different directions.
Added macro `\psGetIsectCenter`
Fixed a bug in the termination of the iteration procedure.
Fixed a bug in the point order of Bézier curves, which was related to a now fixed bug in `pst-func`.
Several other improvements.

0.1 2014-02-19

First CTAN version