

PSTricks

pst-optexp

Intersecting arbitrary curves

v0.1

2014/02/19

Package author:
Christoph Bersch

Contents

1. Introduction	3
1.1. About the package	3
1.2. Requirements	3
1.3. Distribution and installation	3
1.4. License	4
1.5. Acknowledgements	4
2. Usage	5
2.1. Saving paths and curves	5
2.2. Calculating intersections	6
2.3. Visualization of saved paths	7
2.4. Visualization of saved intersections	8
3. Examples	10
A. Revision history	11

1. Introduction

1.1. About the package

`pst-intersect` is a PSTricks package to calculate the intersections of Bezier curves and arbitrary Postscript paths.

1.2. Requirements

`pst-intersect` requires recent versions of `pstricks`, `pst-node`, and `pst-func`.

All PSTricks package rely heavily on the Postscript language so that the typical workflow involves `latex`, `dvips`, and `ps2pdf`. Of course there are several alternative ways to compile your documents.¹

1.3. Distribution and installation

This package is available on CTAN².

The `pst-intersect` package consists of the two main files `pst-intersect.ins` and `pst-intersect.dtx`. By running `tex pst-intersect.ins` the following derived files are generated:

- `pst-intersect.pro`: the Postscript prolog file
- `pst-intersect.sty`: the L^AT_EX style file
- `pst-intersect.tex`: the T_EX file

¹<http://tug.org/PSTricks/main.cgi?file=pdf/pdfoutput>

²<http://mirror.ctan.org/help/Catalogue/entries/pst-intersect.html>

Save the files in a directory which is part of your local $\text{T}_{\text{E}}\text{X}$ tree.

Do not forget to run `texhash` to update this tree. For $\text{MiK}_{\text{T}}\text{E}_{\text{X}}$ users, do not forget to update the file name database (FNDB).

For more detailed information see the documentation of your personal $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ distribution on installing packages to your local $\text{T}_{\text{E}}\text{X}$ system.

1.4. License

Permission is granted to copy, distribute and/or modify this software under the terms of the $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ Project Public License, version 1.3c.³ This package is author-maintained.

1.5. Acknowledgements

I thank Marco Cecchetti, for his `lib2geom-library`⁴ from which I derived great parts of the Postscript code for the Bézier clipping algorithm. Also I want to thank William A. Casselman for the Postscript code of the quicksort procedure and the procedure for calculating the convex hull from his book “Mathematical Illustration”⁵, and the permission to use it. The documentation style is a mixture of the `pst-doc` class (Herbert Voß) and the `ltxdockit` package for the `bi-latex` documentation (Philipp Lehman).

³<http://www.latex-project.org/lppl.txt>

⁴<http://lib2geom.sourceforge.net/>

⁵<http://www.math.ubc.ca/~cass/graphics/text/www/>

2. Usage

The `pst-intersect` package can compute the intersections of arbitrary Postscript paths. These are composed of three primitive operations: lines (`lineto`), third order Bézier curves (`curveto`) and jumps (`moveto`). More specialized constructions, like circles, are converted internally to `curveto` operations. Besides these three path operations, the `pst-intersect` supports Bézier curves up to ninth order. As these aren't primitive Postscript path elements, they require separate handling.

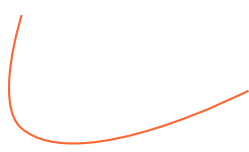
The general workflow consists in defining and saving paths and curves, and then compute the intersections between them. Following, those intersection points can be used as normal PSTricks nodes, or portions of the curves and paths can be retraced (e.g. between two intersections).

2.1. Saving paths and curves

`\pssavepath`[$\langle options \rangle$]{ $\langle curvename \rangle$ }{ $\langle commands \rangle$ }

Saves the complete path, which is generated by $\langle commands \rangle$, under the name $\langle curvename \rangle$. The macro is a modification of `\pscustom`, and does, therefore, supports only the same commands.

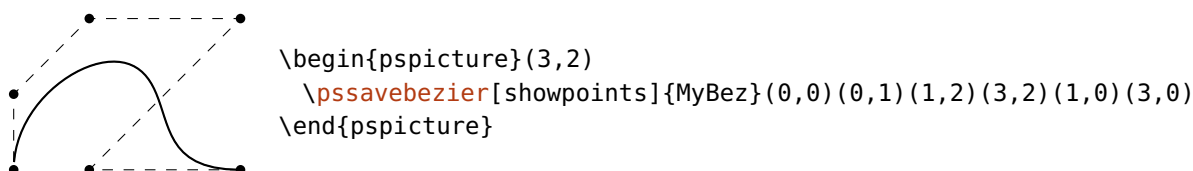
By default, the path is also drawn, which can be changed over the $\langle options \rangle$, e.g. with `linestyle=none`.



```
\begin{pspicture}(3,2)
  \pssavepath[linecolor=D0range]{MyPath}{%
    \pscurve(0,2)(0,0.5)(3,1)
  }%
\end{pspicture}
```

`\pssavebezier`[*<options>*]{*<curvename>*}(*<X₀>*)...(*<X_n>*)

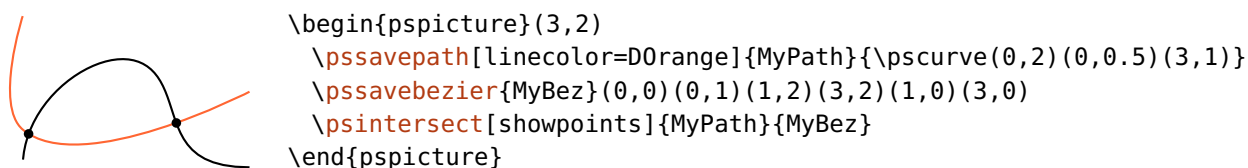
The Postscript language supports only third-order Bézier curves. With the macro `\pssavebezier` you can define Bézier curves up to ninth order. The specified nodes are the control points of the curve, for an n -th order curve $n + 1$ control points are required. The drawing of the curve is done with the `\psBezier` macro from the `pst-func` package.



2.2. Calculating intersections

`\psintersect`{*<curveA>*}{*<curveB>*}

After having saved some paths and curves, you can now calculate the intersections. That is done with the `\psintersect` macro. This needs as arguments two names of paths or curves (the *<curvename>* argument of the two `\pssave*` macros).



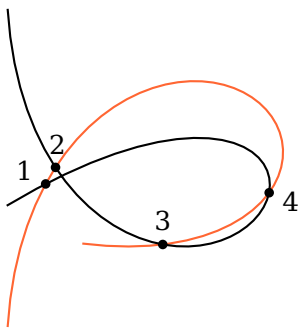
The `showpoints` PSTricks parameter determines, if the intersections are drawn directly.

`name`=*<string>* default: @tmp

The calculated intersections can be saved for later use under this name (see Sec. 2.4).

`saveintersections`=true, false default: true

If this option is set, the intersections are saved as PSTricks nodes with the names *<name>*1, *<name>*2 The numbering is ascending according to the value of their x -coordinate.

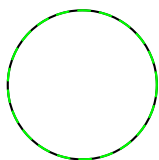


```
\begin{pspicture}(5,5)
  \pssavebezier[linecolor=D0orange]{A}%
    (0,0)(0,5)(5,5)(5,1)(1,1.5)
  \pssavebezier{B}(0,5)(0,0)(5,0)(5,5)(0,2)
  \psintersect[name=C, showpoints]{A}{B}
  \uput[150](C1){1}
  \uput[85](C2){2}
  \uput[90](C3){3}
  \uput[-20](C4){4}
\end{pspicture}
```

2.3. Visualization of saved paths

`\pstracecurve`[*<options>*]{*<curvename>*}

Saved paths and curves can be drawn again with this macro.

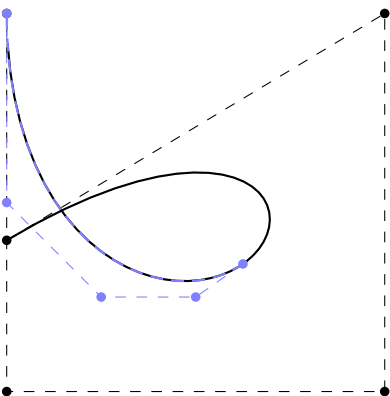


```
\begin{pspicture}(2,2)
  \pssavepath{Circle}{\pscircle(1,1){1}}
  \pstracecurve[linestyle=dashed, linecolor=green]{Circle}
\end{pspicture}
```

`tstart`=*<num>*

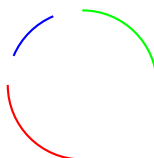
`tstop`=*<num>*

With these parameters also parts of paths and curves can be drawn. For Bézier curves the allowed range is $[0, 1]$, where 0 corresponds to the start of the curve, which is given by the first node given to `\pssavebezier`.



```
\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}
```

Paths can be composed of more than one segment, and the range is $[0, n]$, where n is the number of path segments. For this you must keep in mind, that also e.g. `\pscurve` paths, circles or arcs consist of several segments.



```
\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}
```

Please note, that as of version 0.1 the order of `tstart` and `tstop` doesn't play a role. But for upcoming versions it is planned that `tstart > tstop` reverses the path direction. At the moment arrows aren't supported either.

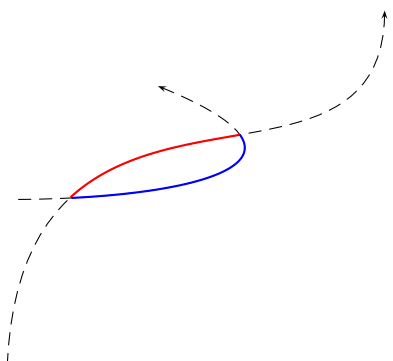
2.4. Visualization of saved intersections

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

```
istart=⟨num⟩
```

```
istop=⟨num⟩
```

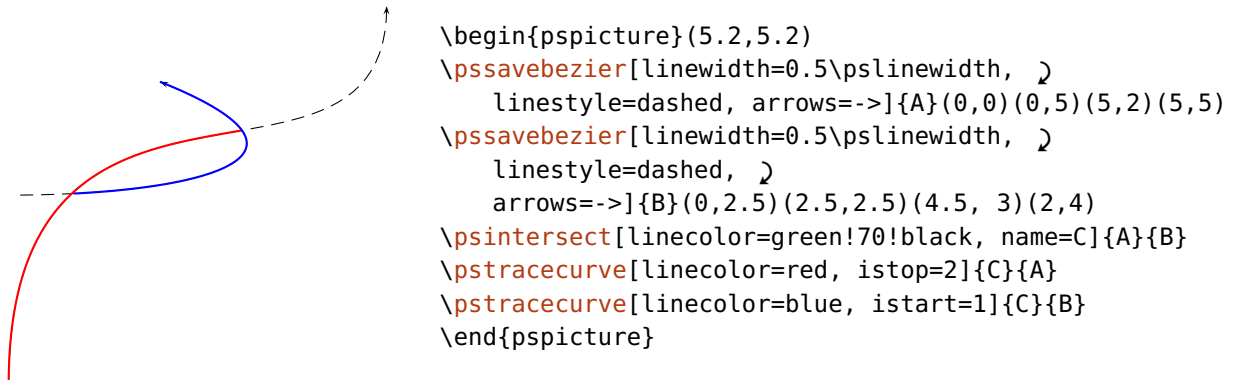
These two parameters can be used to draw path or curve segments between intersections. The intersections are numbered starting at 1 in ascending order along the curve.



```
\begin{pspicture}(5.2,5.2)
  \pssavebezier[linewidth=0.5\pslinewidth, 2
    linestyle=dashed, arrows=->]{A}{(0,0)(0,5)(5,2)(5,5)}
  \pssavebezier[linewidth=0.5\pslinewidth, 2
    linestyle=dashed, 2
    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}
```

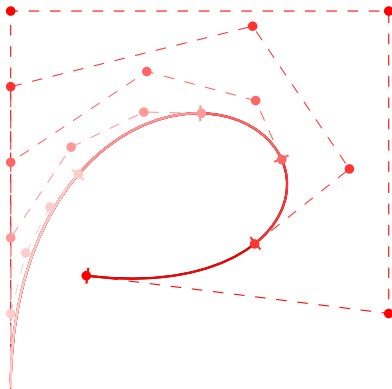
If only one value is specified, e.g. `istop`, the curve is drawn from the start to the respective intersection. If only `istart` is given, the curve is drawn from

this intersection to the curve end. The parameters `istart` and `istop` can be combined with `tstart` and `tstop`.



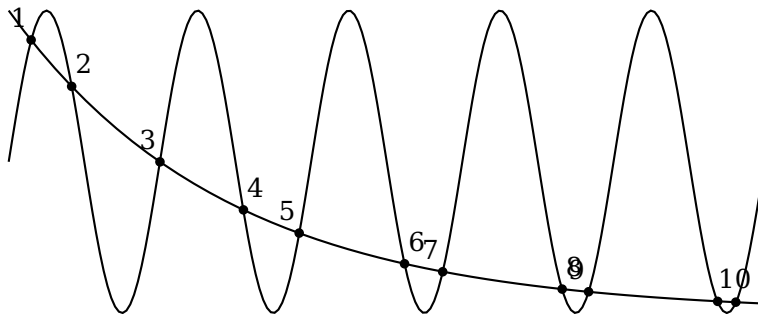
3. Examples

Ex. 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, 2
      arrows=-|, showpoints]{A}
  }%
\end{pspicture}
```

Ex. 3.2: The package can also calculate the intersections of functions which are drawn with `\psplot`. Here you must keep in mind, that such curves consists of plotpoints segments, which must all be considered for intersections, what can result in long calculations.



```
\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+2,\ii=2+2}{5}{\uput[120](C\i){\i}\uput[60](C\ii){\ii}}
\end{pspicture}
```

A. Revision history

This revision history is a list of changes relevant to users of this package. Changes of a more technical nature which do not affect the user interface or the behavior of the package are not included in the list. If an entry in the revision history states that a feature has been *improved* or *extended*, this indicates a modification which either does not affect the syntax and behavior of the package or is syntactically backwards compatible (such as the addition of an optional argument to an existing command). Entries stating that a feature has been *deprecated*, *modified*, *fixed*, *renamed*, or *removed* demand attention. They indicate a modification which may require changes to existing documents.

0.1 2014-02-19

First CTAN version