pst-bezier

A PSTricks package for drawing Bezier curves; v.0.03

Jean-Paul Bécar
Lionel Garnier
Manuel Luque
Tobias Nähring
Herbert Voß

April 12, 2023

Contents

1 Introduction 3

2 Installation and usage of pst-bezier.tex 3

3 The \psbcurve macro 3
3.1 Things that do not work ('known bugs') 8

4 Bezier curve with weighted points 9
4.1 Mathemathical background 9
4.2 Syntax 10
4.3 Three weighted orthogonal points 10
4.4 Half-ellipse 11
4.5 Half-parabola 12
4.6 Branch of a hyperbola 13
4.7 Parabola 13
4.8 Ellipse 14
4.9 Complete circle 14
4.10 Animations 16

5 List of all optional arguments for pst-bezier 18

References 18

The `pstricks` package provides (essentially) two main macros for drawing curves: `\pscurve` and `\psbezier`. Both macros employ Bezier splines.

The `\pscurve` macro takes multiple interpolated points as arguments. Thus, it is easy to draw long multiply bent curves. The problem with `\pscurve` is that there is no easy way to change the automatically computed control points without simultaneously changing the interpolated points. Note that some control is possible via the `curvature` option.

The `\psbcurve` macro gives full control over the interpolation points and the control points of one Bezier polynominal of degree three (two interpolated points and two control points).

Thanks to:
Jean-Côme Charpentier.

1 Introduction

If one demands for the access to certain control points of one multiply bent curve one has to use multiple instances of the `\psbezier` macro. With this approache each inner interpolation point of the curve has to be input twice. Furthermore, if one needs smooth joints one has to compute control points symmetrically to the corresponding interpolation points for every joint even if one does not care so much about the exact tangential direction at some of those joints. That can be rather tedious.

The `\psbcurve` macro of the package `pst-bezier` is intented to demonstrate a way to combine the nice properties of the macros `\pscurve` and `\psbezier`. It provides an easy input format to describe ‘arbitrarily’ many interpolation points of a curve and to fix the control points at some freely selected interpolation points.

Note, that `pst-bezier` is *no final package* (e.g. the automatical computation of the control points is not as refined as that one for the macro `\pscurve`).

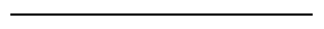
2 Installation and usage of `pst-bezier.tex`

Installation: As prerequisites for `pst-bezier` you need resent working versions of LaTeX and `pstricks`. The files `pst-bezier.tex` and `pst-bezier.sty` must be somewhere in your TeX-input path. Further more, the file `pst-bezier.pro` must be in some path, where `dvips` can find it.

Usage: As usual, load the packages `pstricks` and `pst-bezier` in that order via the `\usepackage` macro.

Now you are ready to use the `\psbcurve` macro within your document body. This macro is described in the next section with all its options.

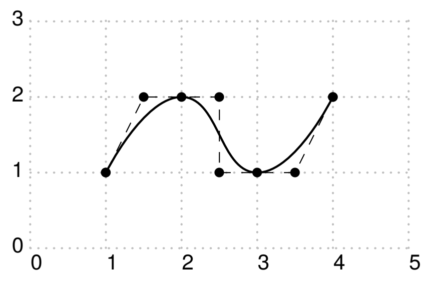
Whith the following simple LaTeX-source code you can test whether you have correctly installed the package:



```
\documentclass{minimal}
\usepackage{pstricks}
\usepackage{pst-bezier}
\begin{document}
 \begin{pspicture}(0,-0.4)(6,2)
   \psbcurve(1,2)(5,2) % Draw just one straight line.
 \end{pspicture}
\end{document}
```

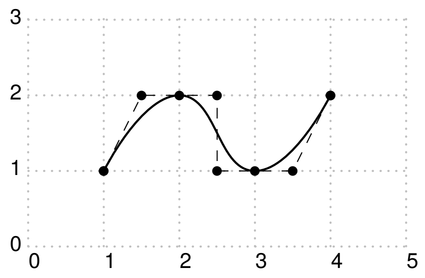
3 The `\psbcurve` macro

In the most simple form you can specify any number of interpolation points as the argument of `\psbcurve`.



```
\begin{pspicture}[showgrid](0,-0.4)(5,3)
 \psbcurve[showpoints](1,1)(2,2)(3,1)(4,2)
\end{pspicture}
```

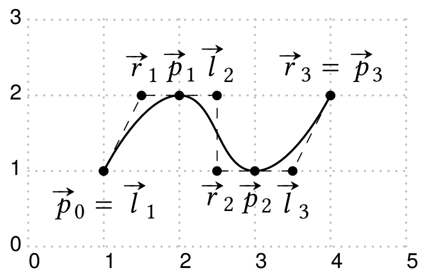
As usual, options can be specified within brackets.



```
\begin{pspicture}[showgrid](0,-0.4)(5,3)
  \psbcurve[showpoints](1,1)(2,2)(3,1)(4,2)
\end{pspicture}
```

As you can see in the above example, the `showpoints` feature works (partially) with `\psbcurve`.

The next figure shows again the curve from the first example. This time labels are added to the points (this is just for the following description, it is not a feature of `\psbcurve`).



```
\begin{pspicture}[showgrid](0,-0.4)(5,3)
  \psbcurve[showpoints](1,1)(2,2)(3,1)(4,2)
  \uput[-90](1,1){$\vec{p}_{0}=\vec{l}_{1}$}
  \uput[90](1.5,2){$\vec{r}_{1}$}
  \uput[90](2,2){$\vec{p}_{1}$}
  \uput[90](2.5,2){$\vec{l}_{2}$}
  \uput[-90](2.5,1){$\vec{r}_{2}$}
  \uput[-90](3,1){$\vec{p}_{2}$}
  \uput[-90](3.5,1){$\vec{l}_{3}$}
  \uput[90](4,2){$\vec{r}_{3}=\vec{p}_{3}$}
\end{pspicture}
```

The points labeled with $\vec{p}_k$ ($k = 0, \dots, 3$) are the interpolation points, these ones labelled with $\vec{l}_1, \dots, \vec{l}_3$, and these ones labelled with $\vec{r}_1, \dots, \vec{r}_3$ are the left and right control points, respectively.

Between each consecutive pair $\vec{p}_{k-1}, \vec{p}_k$ of interpolation points the `\psbcurve` macro draws a cubic Bezier spline. The control points $\vec{l}_k$ and $\vec{r}_k$ determine the tangential direction of the bezier spline at the interpolation points. More exactly, the bezier spline from $\vec{p}_{k-1}$ to $\vec{p}_k$ is tangent to the vector $\vec{l}_k - \vec{p}_{k-1}$ at the point $\vec{p}_{k-1}$ and tantengial to the vektor $\vec{r}_k - \vec{p}_k$ at the point $\vec{p}_k$.

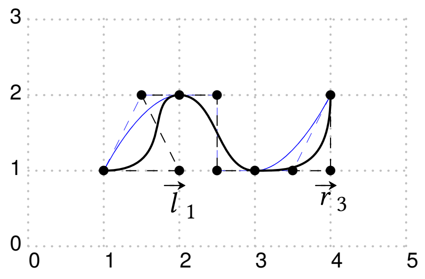
Without any optional modifier arguments (described later in this text) the control points are computed automatically from the interpolation points by the formulas[1]

$$\begin{aligned}
\vec{l}_1 &= \vec{p}_0 & \\
\vec{l}_k &= t_k(\vec{p}_k - \vec{p}_{k-2}) & \text{for } k = 2, \dots, n \\
\vec{r}_k &= t_k(\vec{p}_{k-1} - \vec{p}_{k+1}) & \text{for } k = 1, \dots, n-1 \\
\vec{r}_n &= \vec{p}_n &
\end{aligned}$$

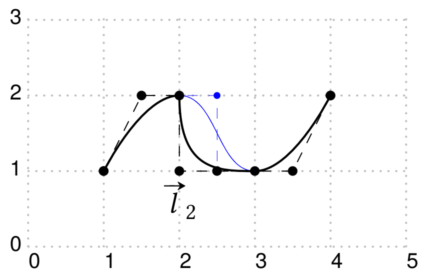
1 Note that this method is very crude. To compute the curve such that the curvature is continuous would require solving a nonlinear system of equations. That is not implemented yet.

where t_k $(k = 1, \dots, n)$ are real coefficients which are called tension and which default to the value `bcurveTension=0.25`.

You can change the appearance of the curve by several modifiers. First of all you can directly set the left and right control points via the modifiers `l(`x,y`)` and `r(`x,y`)`, resp., as shown in the next two examples. The unmodified curve is drawn in the background in blue color.

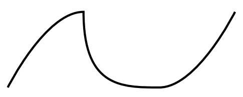


```
\pspicture[showgrid](0,-0.4)(5,3)
\psset{showpoints}
\psbcurve[linecolor=blue,linewidth=0.01](1,1)%
  (2,2)(3,1)(4,2)
\psbcurve(1,1)l(2,1)(2,2)(3,1)r(4,1)(4,2)
\uput[-90](2,1){$\vec{l}_{1}$}
\uput[-90](4,1){$\vec{r}_{3}$}
\endpspicture
```



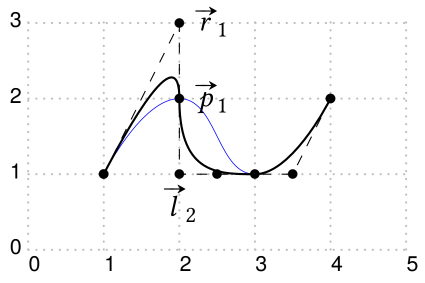
```
\pspicture[showgrid](0,-0.4)(5,3)
\psset{showpoints}
\psbcurve[linecolor=blue,linewidth=0.01](1,1)%
  (2,2)(3,1)(4,2)
\psbcurve(1,1)(2,2)l(2,1)(3,1)(4,2)
\uput[-90](2,1){$\vec{l}_{2}$}
\endpspicture
```

On the right hand side the last example is shown once more without grid and with `showpoints=false` . There, you see that there is a corner at the second interpolation point.



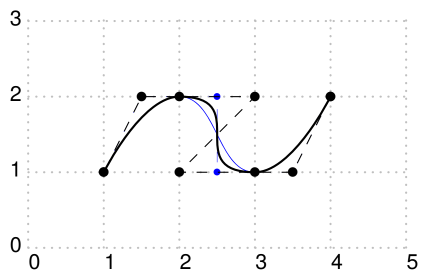
```
\pspicture(0,-0.4)(5,3)
\psbcurve(1,1)(2,2)l(2,1)(3,1)(4,2)
\endpspicture
```

If you change some left control point $\vec{l}_k$ with the help of the `L(`x,y`)` modifier then the control point $\vec{r}_{k-1}$ is set symmetrically to $\vec{l}_k$ with respect to the interpolation point $\vec{p}_{k-1}$. In that way you get a smooth joint as demonstrated in the next example.



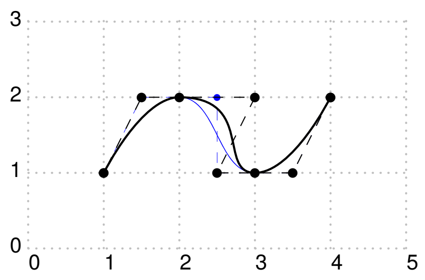
```
\pspicture[showgrid](0,-0.4)(5,3)
\psbcurve[linecolor=blue,linewidth=0.01](1,1)%
  (2,2)(3,1)(4,2)
\psset{showpoints}
\psbcurve(1,1)(2,2)L(2,1)(3,1)(4,2)
\uput[-90](2,1){$\vec{l}_{2}$}
\uput[0](2,2){$\vec{p}_{1}$}
\uput[0](2,3){$\vec{r}_{1}$}
\endpspicture
```

With the t{t} modifier you can change the tension of the automatically computed control points of the current Bezier spline.

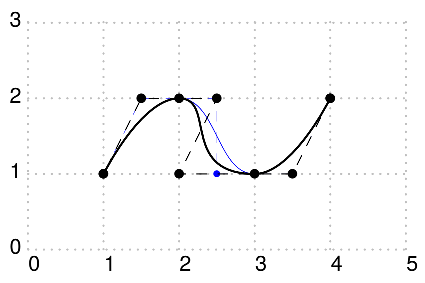


```
\pspicture[showgrid](0,-0.4)(5,3)
\psset{showpoints}
\psbcurve[linecolor=blue,linewidth=0.01](1,1)%
  (2,2)(3,1)(4,2)
\psbcurve(1,1)(2,2)t{0.5}(3,1)(4,2)
\endpspicture
```

As you can see from the example both control points of the current spline are affected by the t{t} modifier. If you want to change the tension of just the left or right control point you can use the tl{t} or tr{t} modifier, respectively, as demonstrated in the following two examples.

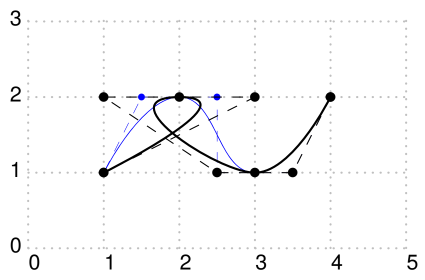


```
\pspicture[showgrid](0,-0.4)(5,3)
\psset{showpoints}
\psbcurve[linecolor=blue,linewidth=0.01](1,1)%
  (2,2)(3,1)(4,2)
\psbcurve(1,1)%
  (2,2)tl{0.5}(3,1)(4,2)
\endpspicture
```



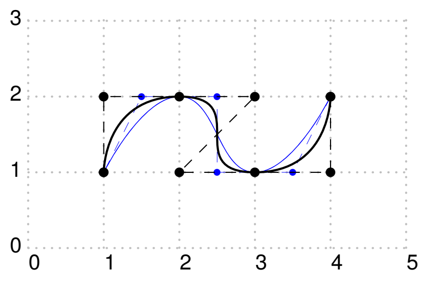
```
\pspicture[showgrid](0,-0.4)(5,3)
\psset{showpoints}
\psbcurve[linecolor=blue,linewidth=0.01](1,1)%
  (2,2)(3,1)(4,2)
\psbcurve(1,1)(2,2)tr{0.5}(3,1)(4,2)
\endpspicture
```

The ts{*t*} modifier changes the tension of the left and right control points next to the interpolation point which stands in front of the modifier. In the next example a negative tension value leads to a rather surprising effect.



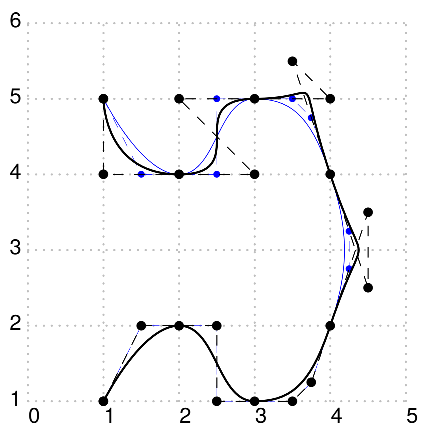
```
\pspicture[showgrid](0,-0.4)(5,3)
\psset{showpoints}
\psbcurve[linecolor=blue,linewidth=0.01](1,1)%
  (2,2)(3,1)(4,2)
\psbcurve(1,1)(2,2)ts{-0.5}(3,1)(4,2)
\endpspicture
```

The default value of the tension can be set with the option bcurveTension as in the following example.



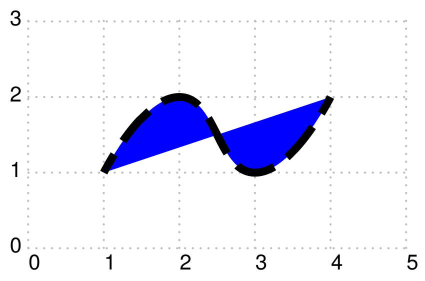
```
\pspicture[showgrid](0,-0.4)(5,3)
\psset{showpoints}
\psbcurve[linecolor=blue,linewidth=0.01](1,1)%
  (2,2)(3,1)(4,2)
\psbcurve[bcurveTension=0.5](1,1)%
  (2,2)(3,1)(4,2)
\endpspicture
```

You can set this option also with the help of the \psset macro. It is even possible to change the value of bcurveTension in the middle of a \psbcurve. Just use the modifier T{*t*} for that purpose as shown in the following example.



```
\pspicture[showgrid](0,0.6)(5,6)
\psset{showpoints}
\psbcurve[linecolor=blue,linewidth=0.01](1,1)%
  (2,2)(3,1)(4,2)(4,4)(3,5)%
  (2,4)(1,5)
\psbcurve(1,1)(2,2)(3,1)(4,2)%
  T{0.5}(4,4)(3,5)(2,4)(1,5)
\endpspicture
```

Certainly, you can use the `T{`*t*`}` modifier several times in one curve. (Try it for yourself.) The `linestyle` and `fillstyle` options (and several more) are respected by `\psbcurve` as the following example shows.



```
\pspicture[showgrid](0,-0.4)(5,3)
\psbcurve[linestyle=dashed,
  linewidth=3pt,
  dash=0.5 0.2,
  fillstyle=solid,
  fillcolor=blue](1,1)(2,2)(3,1)(4,2)
\endpspicture
```

3.1 Things that do not work ('known bugs')

As already mentioned this project is something like an experiment. So, there are many things that do not work.

- new lines inside the argument list are not ignored.
- The control points are computed in a rather crude way (see above). The `curvature` option is not recognised.
- If `fillstyle` is set to `solid` and `showpoints` then the fill color covers the interpolation and control points.
- arrow heads do not work.

4 Bezier curve with weighted points

4.1 Mathemathical background

A mass point is a weighted point $(P;\omega)$ with $\omega \neq 0$ or a vector $\left(\vec{P};0\right)$ with a weight equal to 0. A generic mass point is noted $(P;\omega)$.

Using the quadratic Bernstein polynomials, a rational quadratic Bézier curve having three control mass points $(P_0;\omega_0)$, $(P_1;\omega_1)$ and $(P_2;\omega_2)$, is defined as follow:

Definition 1 *: Rational quadratic Bézier curve (BR curve)*

Let ω_0, ω_1 and ω_2 be three real numbers. Let $(P_0;\omega_0)$, $(P_1;\omega_1)$ and $(P_2;\omega_2)$ be three mass points, these points are not collinear.

Define two sets $I = \{i|\omega_i \neq 0\}$ and $J = \{i|\omega_i = 0\}$

Define the function ω_f from $[0;1]$ to $\mathbb{R}$ as follows

$$\omega_f(t) = \sum_{i\in I} \omega_i \times B_i(t) \tag{1}$$

A mass point $(M;\omega)$ or $\left(\vec{u};0\right)$ belongs to the quadratic Bézier curve defined by the three control mass points $(P_0;\omega_0)$, $(P_1;\omega_1)$ and $(P_2;\omega_2)$, if there is a real t_0 in $[0;1]$ such that:

- *if $\omega_f(t_0) \neq 0$ then we have*

$$\overrightarrow{OM} = \frac{1}{\omega_f(t_0)}\left(\sum_{i\in I}\omega_i B_i(t_0)\overrightarrow{OP_i}\right) + \frac{1}{\omega_f(t_0)}\left(\sum_{i\in J} B_i(t_0)\vec{P_i}\right) \tag{2}$$

- *if $\omega_f(t_0) = 0$ then we have*

$$\vec{u} = \sum_{i\in I}\omega_i B_i(t_0)\overrightarrow{OP_i} + \sum_{i\in J} B_i(t_0)\vec{P_i} \tag{3}$$

The reduced discriminant of the denominator $\omega_f(t_0)$ is

$$\Delta' = \omega_1^2 - \omega_2\omega_0 \tag{4}$$

and we can state the following fundamental result:

- ⋆ if $\omega_1^2 - \omega_2\omega_0 = 0$ then the denominator has one and only one root, the curve is a parabolic arc;
- ⋆ if $\omega_1^2 - \omega_2\omega_0 > 0$ then the denominator has two distinct roots, the curve is a hyperbolic arc;
- ⋆ if $\omega_1^2 - \omega_2\omega_0 < 0$ then the denominator does not vanish, the curve is an elliptical arc.

We can note w.l.o.g.[2] that one of the weights can be equal to 1. If ω_0 is not equal to 0, we choose $\omega_0 = 1$, else, we choose $\omega_1 = 1$, and we can characterise the type of the conic from the mass points of the BR curve, see Table 1.

From the access rights used by Unix and Linux, we define a bijection f between ${\mathbb{F}_2}^3 - \{(0,0,0)\}$ and the set $\{1,2,3,4,5,6,7\}$. From $(\omega_2,\omega_1,\omega_0)$, we define a triplet (b_2,b_1,b_0) as follow: if $w_i \neq 0$ then $b_i = 1$ else $b_i = 0$. Then

$$f(\omega_2,\omega_1,\omega_0) = b_2 \times 4 + b_1 \times 2 + b_0$$

If $f(\omega_2,\omega_1,\omega_0) = 7$, the control points are weighted points: the curve is an elliptical arc, a parabolic arc or a hyperbolic arc. If $(\omega_2,\omega_1,\omega_0) = (1,-1,1)$, the parabolic arc is not bounded and for $t = \frac{1}{2}$, the mass point is

2 We can permute the role of P_0 and P_2

Conic	Three weighted points	Points and vectors
Parabola	$(P_0;1), (P_1;\omega)\ (P_2;\omega^2)$	$(P_0;1), \left(\overrightarrow{P_1};0\right)\ \left(\overrightarrow{P_2};0\right)$
Ellipse	$(P_0;1), (P_1;\omega_1), (P_2;\omega_2), \omega_2 > \omega_1^2$	$(P_0;1), \left(\overrightarrow{P_1};0\right)\ (P_2;1)$
Hyperbola	$(P_0;1), (P_1;\omega_1)\ (P_2;\omega_2), \omega_2 < \omega_1^2$	$(P_0;1), \left(\overrightarrow{P_1};0\right)\ (P_2;-1)$
		$\left(\overrightarrow{P_0};0\right), (P_1;1)$ and $\left(\overrightarrow{P_2};0\right)$

Table 1: Types of conics defined by Bézier curves with control mass points.

a direction vector of the parabola axis. If $(\omega_2,\omega_1,\omega_0) = (1,-2,1)$, the hyperbolic arc is not bounded and there exists t in $]0,1[$ such as the mass point is a direction vector of one of the asymptotes of the hyperbola.
If $f(\omega_2,\omega_1,\omega_0) = 1$, the first control point is a weighted point, the others are vectors: the curve is a parabolic arc. The Bézier curve is defined by

$$\begin{cases} \dfrac{1}{\omega_0\, B_0\,(t_0)}\left(\omega_0\, B_0\,(t_0)\,\overrightarrow{OP_0} + B_1\,(t_0)\,\overrightarrow{P_1} + B_2\,(t_0)\,\overrightarrow{P_2}\right) & \text{if } t_0 \in [0,1[\\ \overrightarrow{P_2} & \text{if } t_0 = 1 \end{cases} \tag{5}$$

If $f(\omega_2,\omega_1,\omega_0) = 4$, the Bézier curve can be defined in the same way.
If $f(\omega_2,\omega_1,\omega_0) = 2$, the intermediate control point is a weighted point, the others are vectors: the curve is a branch of a hyperbola. The Bézier curve is defined by

$$\begin{cases} \dfrac{1}{\omega_1\, B_1\,(t_0)}\left(\omega_1\, B_1\,(t_0)\,\overrightarrow{OP_1} + B_0\,(t_0)\,\overrightarrow{P_0} + B_2\,(t_0)\,\overrightarrow{P_2}\right) & \text{if } t_0 \in]0,1[\\ \overrightarrow{P_0} & \text{if } t_0 = 0 \\ \overrightarrow{P_2} & \text{if } t_0 = 1 \end{cases} \tag{6}$$

and the centre of the hyperbola is P_1. The vector $\overrightarrow{P_0}$ is a direction vector of an asymptote of the hyperbola whereas the vector $\overrightarrow{P_2}$ is a direction vector of the other asymptote.
If $f(\omega_2,\omega_1,\omega_0) = 5$, the intermediate control point is a vector, the others are weighted points: the curve is an elliptical arc. The Bézier curve is defined by

$$\frac{1}{\omega_0\, B_0\,(t_0) + \omega_2\, B_2\,(t_0)}\left(\omega_0\, B_0\,(t_0)\,\overrightarrow{OP_0} + B_1\,(t_0)\,\overrightarrow{P_1} + \omega_2\, B_2\,(t_0)\,\overrightarrow{OP_2}\right),\quad t_0 \in [0,1] \tag{7}$$

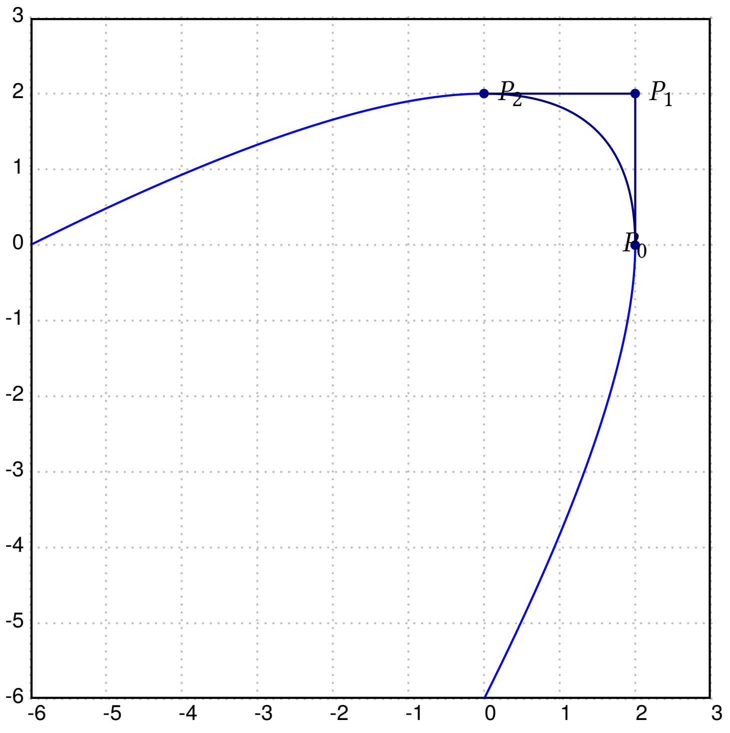
and the tangent vector to the curve at P_0 or P_2 is parallel to $\overrightarrow{P_1}$.

4.2 Syntax

```
\psRQBCmasse [Options] (x0,y0)(x1,y1)(x2,y2){w0,w1,w2}
```

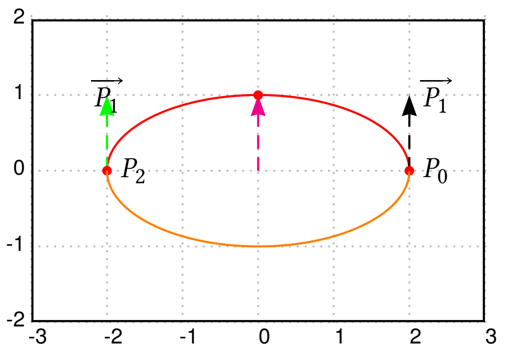
For the coordinates of the points all possible kinds of coordinates are possible, like polar, PostScript, nodes, ...

4.3 Three weighted orthogonal points



```
\begin{pspicture}[showgrid](-6,-6.4)(3,3)
\psclip{\psframe(-6,-6)(3,3)}
  \psRQBCmasse[linecolor=blue](2,0)(2,2)(0,2){1,-1,1}
  \psRQBCmasse[linecolor=navy,autoTrace](2,0)(2,2)(0,2){1,1,1}
  \rput(P0){$P_0$}\uput[r](P1){$P_1$}\uput[r](P2){$P_2$}
\endpsclip%
\end{pspicture}
```

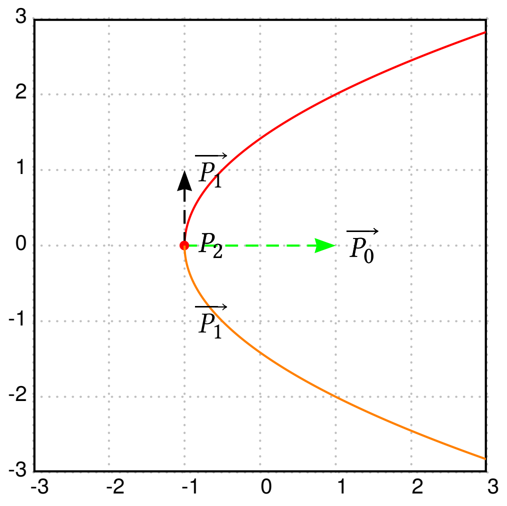
4.4 Half-ellipse



```
\begin{pspicture}[showgrid](-3,-2.4)(3,2)
\psframe(-3,-2)(3,2)
\psRQBCmasse[linecolor=red,autoTrace](2,0)(0,1)(-2,0){1,0,1}
\uput[r](P0P1){$\overrightarrow{P_1}$} \uput[r](P2){$P_2$}
\rput(P1P2){$\overrightarrow{P_{1}}$} \uput[r](P0){$P_0$}
\psRQBCmasse[linecolor=orange,autoTrace=false](2,0)(0,-1)(-2,0){1,0,1}
\end{pspicture}
```

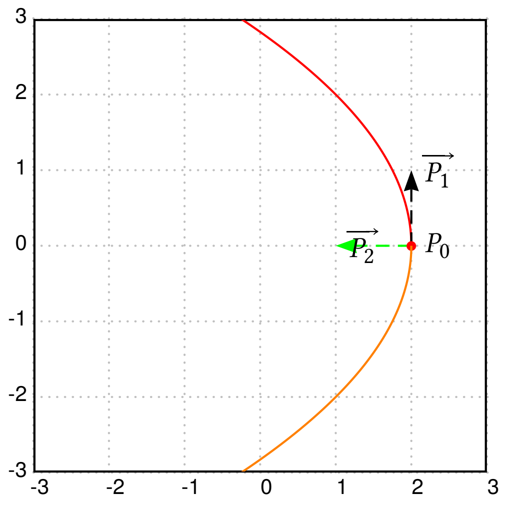
4.5 Half-parabola

Point P_2 and two vectors



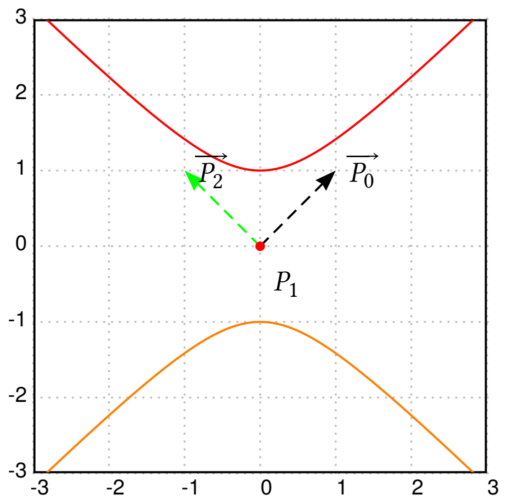
```
\begin{pspicture}[showgrid](-3,-3.4)(3,3)
\psclip{\psframe(-3,-3)(3,3)}
  \psRQBCmasse[linecolor=red,autoTrace](2,0)(0,1)(-1,0){0,0,1}
  \uput[r](P1P2){$\overrightarrow{P_1}$} \uput[r](P2){$P_2$}
  \uput[r](P0P2){$\overrightarrow{P_0}$}
  \psRQBCmasse[linecolor=orange,autoTrace=false](2,0)(0,-1)(-1,0){0,0,1}
  \uput[r](P1P2){$\overrightarrow{P_1}$} \uput[r](P2){$P_2$}
  \uput[r](P0P2){$\overrightarrow{P_0}$}
\endpsclip
\end{pspicture}
```

Point P_0 and two vectors



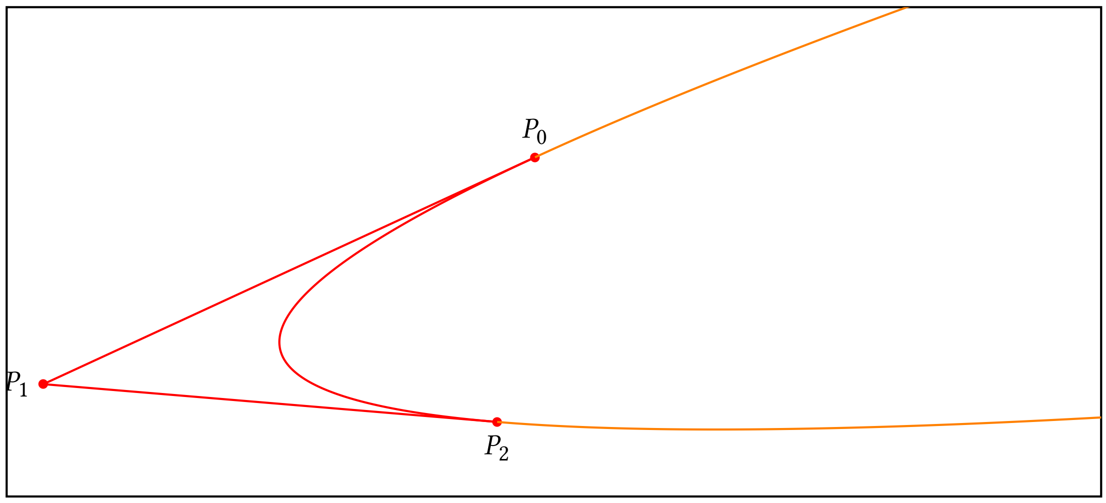
```
\begin{pspicture}[showgrid](-3,-3.4)(3,3)
\psclip{\psframe(-3,-3)(3,3)}
  \psRQBCmasse[linecolor=red,autoTrace](2,0)(0,1)(-1,0){1,0,0}
  \uput[r](P0P1){$\overrightarrow{P_1}$} \uput[r](P0){$P_0$}
  \uput[r](P0P2){$\overrightarrow{P_2}$}
  \psRQBCmasse[linecolor=orange,autoTrace=false](2,0)(0,-1)(-1,0){1,0,0}
\endpsclip%
\end{pspicture}
```

4.6 Branch of a hyperbola



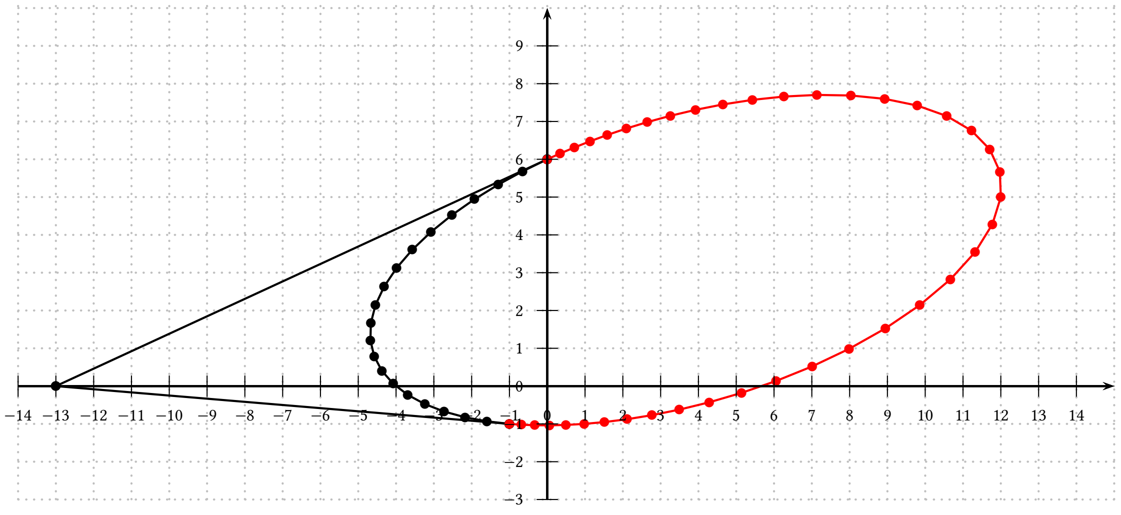
```
\begin{pspicture}[showgrid](-3,-3.4)(3,3)
\psclip{\psframe(-3,-3)(3,3)}
  \psRQBCmasse[linecolor=red,autoTrace](1,1)(0,0)(-1,1){0,1,0}
  \uput[r](P0){$\overrightarrow{P_0}$} \uput[r](0,-0.5){$P_1$}
  \uput[r](P2){$\overrightarrow{P_2}$}
  \psRQBCmasse[linecolor=orange,autoTrace=false](1,1)(0,0)(-1,1){0,-1,0}
\endpsclip%
\end{pspicture}
```

4.7 Parabola



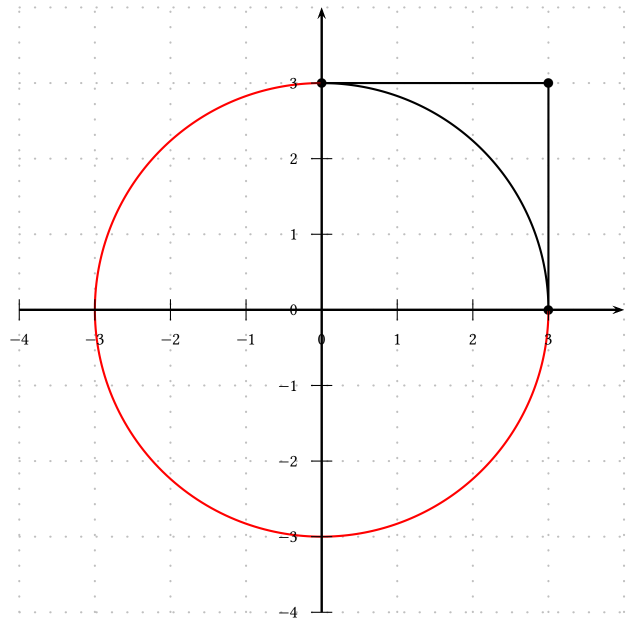
```
\psset{unit=0.5}
\begin{pspicture}(-14,-3.4)(15,10)
\psclip{\psframe(-14,-3)(15,10)}
  \psRQBCmasse[linecolor=red,autoTrace](0,6)(-13,0)(-1,-1){1,1,1}
  \psRQBCmasse[linecolor=orange](0,6)(-13,0)(-1,-1){1,-1,1}
  \uput[u](P0){$P_0$}\uput[l](P1){$P_1$}\uput[d](P2){$P_2$}
\endpsclip
\end{pspicture}
```

4.8 Ellipse

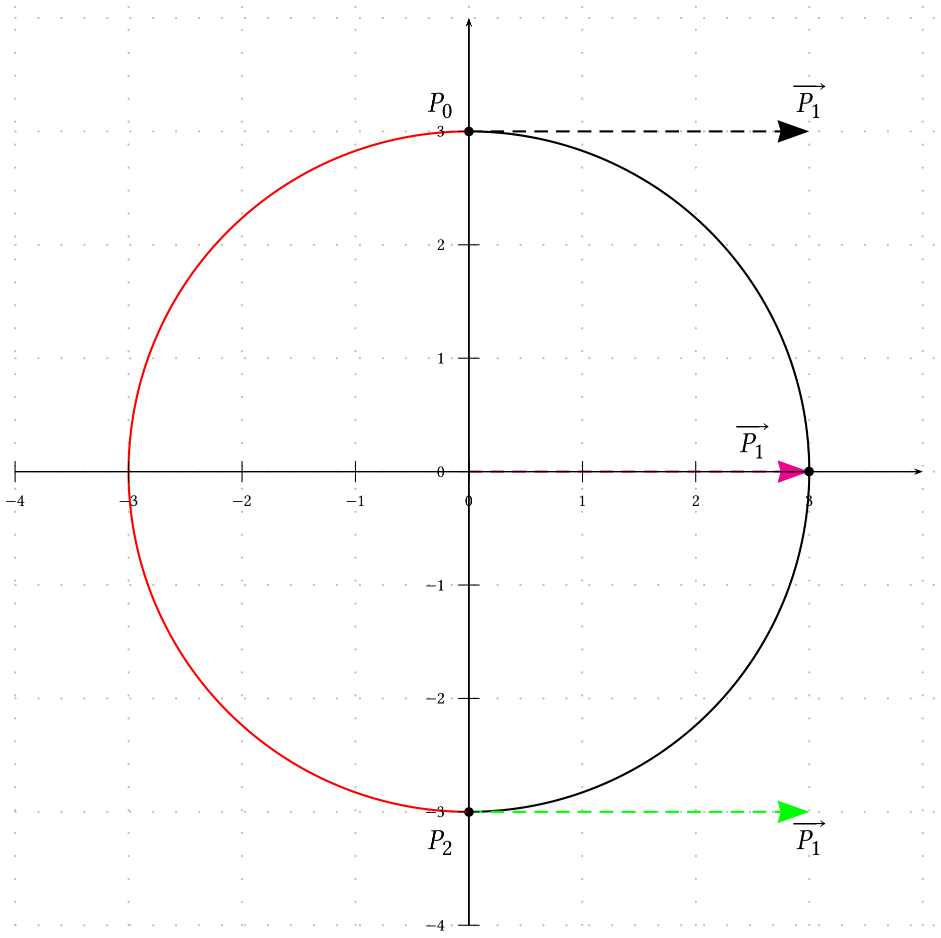


```
\psset{unit=0.5}
\begin{pspicture}(-14,-3.4)(15,10)
\psgrid[subgriddiv=0,gridcolor=lightgray,griddots=5,gridlabels=0pt]
%\psplotImp[linewidth=0.5pt,linecolor=blue,algebraic](-6,-3)(15,10)%
 %{ -0.044*x^2-0.161*y^2 + 0.075*x*y + 0.074*x + 0.797*y + 1}
\psRQBCmasse[nPoints=20,autoTrace,showpoints](0,6)(-13,0)(-1,-1){1,0.5,1}
\psRQBCmasse[nPoints=40,linecolor=red,showpoints](0,6)(-13,0)(-1,-1){1,-0.5,1}
\psaxes[labelFontSize=\scriptscriptstyle]{->}(0,0)(-14,-3)(15,10)
\end{pspicture}
```

4.9 Complete circle



```
\psset{unit=1}
\begin{pspicture}(-4,-4.4)(4,4)
\psgrid[subgriddiv=0,gridcolor=lightgray,griddots=5,gridlabels=0pt]
\psRQBCmasse[autoTrace](0,3)(3,3)(3,0){1,1,2}
\psRQBCmasse[linecolor=red](0,3)(3,3)(3,0){1,-1,2}
\psaxes[labelFontSize=\scriptscriptstyle]{->}(0,0)(-4,-4)(4,4)
\end{pspicture}
```



```
\psset{unit=1.5}
\begin{pspicture}(-4,-4.4)(4,4)
\psgrid[subgriddiv=0,gridcolor=lightgray,griddots=5,gridlabels=0pt]
\psRQBCmasse[autoTrace](0,3)(3,0)(0,-3){1,0,1}
\uput[u](-0.25,3){$P_0$}
\uput[u](-0.25,-3.5){$P_2$}
\uput[u](3,3){$\overrightarrow{P_1}$}
\uput[u](3,-3.5){$\overrightarrow{P_1}$}
\uput[u](2.5,0){$\overrightarrow{P_1}$}
\psRQBCmasse[linecolor=red](0,3)(-3,0)(0,-3){1,0,1}
\psaxes[labelFontSize=\scriptscriptstyle,linewidth=0.01]{->}(0,0)(-4,-4)(4,4)
\end{pspicture}
```

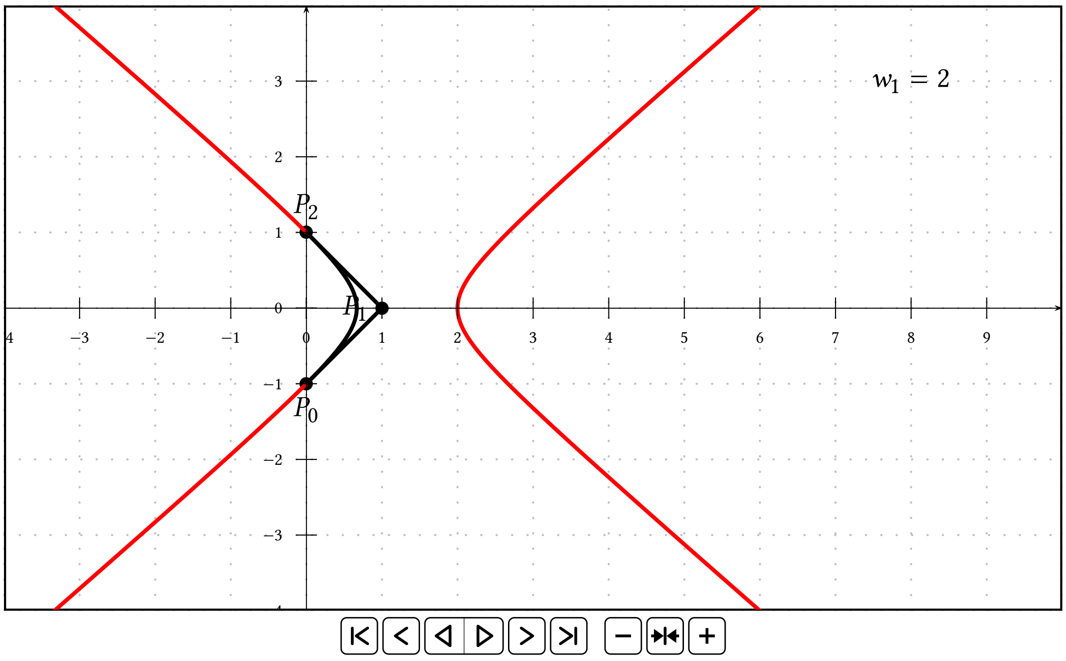
We get a circle because we have

$$\left\{\begin{array}{rcl} \omega_0 \times \omega_2 \times P_0P_2^2 & = & 4 \times \overrightarrow{P_1}^2 \\ \overrightarrow{P_0P_2} & \perp & \overrightarrow{P_1} \end{array}\right. \tag{8}$$

4.10 Animations

$w_0 = 1$, $w_2 = 1$ and a variable w_1

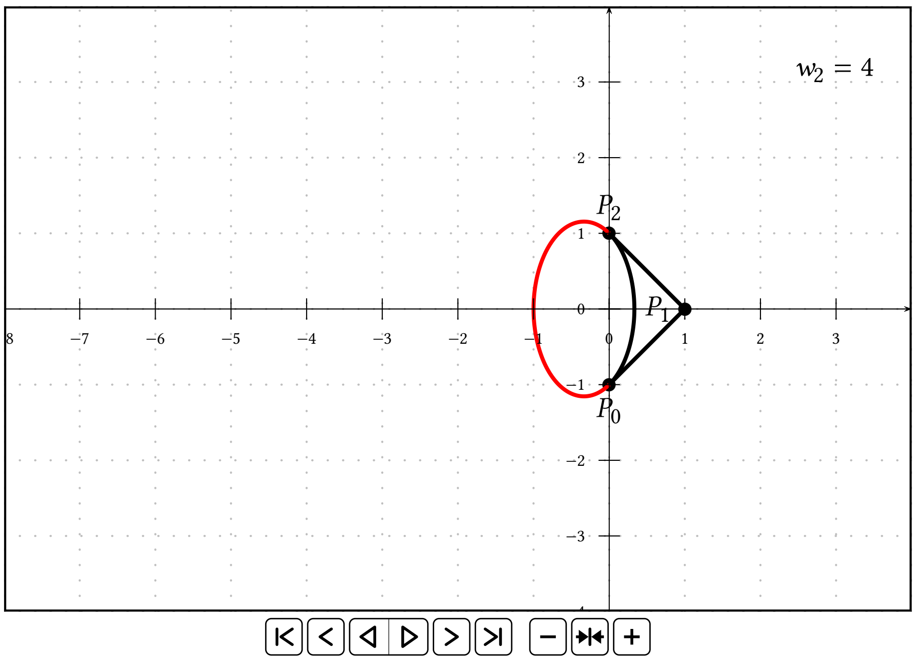
With the beginning of $w_1 = 0$ the curves are swapped. In the case of Bezier curves $w_1 = 0$ gives only the $[P_0P_2]$ segment. Using the mass points, the point P_1 no longer exists but we get the vector $\overrightarrow{P_1}$.



```
\begin{animateinline}[controls,loop,palindrome,
                begin={\begin{pspicture}(-4,-4)(10,4)},
                end={\end{pspicture}}]{3}% 3 images/s
\multiframe{40}{rA=2.0+-0.1,rB=-2.0+0.1}{%
 \psgrid[subgriddiv=0,gridcolor=lightgray,griddots=5,gridlabels=0pt]
 \psclip{\psframe(-4,-4)(10,4)}
   \psRQBCmasse[autoTrace,linewidth=1.5pt](0,-1)(1,0)(0,1){1,\rA,1}
   \uput[u](P2){$P_2$}\uput[l](P1){$P_1$}\uput[d](P0){$P_0$}
   \psRQBCmasse[linecolor=red,linewidth=1.5pt](0,-1)(1,0)(0,1){1,\rB,1}
   \psaxes[labelFontSize=\scriptscriptstyle,linewidth=0.01]{->}(0,0)(-4,-4)(10,4)
   \rput(8,3){$w_1=\rA$}%
 \endpsclip
}
\end{animateinline}
```

$w_0 = 1, |w_1| = 1$ and a variable w_2

The use of $|w_1|$ provides both arcs and the whole cone.



```
\begin{animateinline}[controls,loop,palindrome,
                begin={\begin{pspicture}(-8,-4)(4,4)},
                end={\end{pspicture}}]{3}% 3 images/s
\multiframe{80}{rA=4.0+-0.1}{%
  \psgrid[subgriddiv=0,gridcolor=lightgray,griddots=5,gridlabels=0pt]
  \psclip{\psframe(-8,-4)(4,4)}
    \psRQBCmasse[autoTrace,linewidth=1.5pt](0,-1)(1,0)(0,1){1,1,\rA}
    \uput[u](P2){$P_2$}\uput[l](P1){$P_1$}\uput[d](P0){$P_0$}
    \psRQBCmasse[linecolor=red,linewidth=1.5pt](0,-1)(1,0)(0,1){1,-1,\rA}
    \psaxes[labelFontSize=\scriptscriptstyle,linewidth=0.01]{->}(0,0)(-8,-4)(4,4)
    \rput[rb](3.5,3){$w_2=\rA$}%
  \endpsclip
}
\end{animateinline}
```

5 List of all optional arguments for `pst-bezier`

Key	Type	Default
`bcurveTension`	`ordinary`	`0.25`
`nPoints`	`ordinary`	`[none]`
`showPolygon`	`boolean`	`true`
`autoTrace`	`boolean`	`true`

References

[1] Jean-Paul Bécar. “Forme (BR) des coniques et de leurs faisceaux”. phdthesis. Valenciennes, France: Université de Valenciennes et de Hainaut-Cambrésis, LIMAV, 12 **december** 1997.

[2] Lionel Garnier. *Courbes de Bézier et coniques.* URL: `http://ufrsciencestech.u-bourgogne.fr/~garnier/Migs/03_CourbesBezierPointsMassiquesEleve.pdf` (**urlseen** 20/08/2016).

[3] Lionel Garnier **and** Jean-Paul Bécar. “Mass points, Bézier curves and conics: a survey”. **in***Eleventh International Workshop on Automated Deduction in Geometry*: Proceedings of ADG 2016. Strasbourg, France, **june** 2016, **pages** 97–116. URL: `http://ufrsciencestech.u-bourgogne.fr/~garnier/publications/adg2016/` (**urlseen** 20/08/2016).

[4] Lionel Garnier, Jean-Paul Bécar **and** Lucie Drouton. *Surfaces canal et courbes de Bézier rationnelles quadratiques.* URL: `http://ufrsciencestech.u-bourgogne.fr/~garnier/publications/hippocampe/64_GTMG2016_courbesBezierSurfacesCanal.pdf` (**urlseen** 20/08/2016).

[5] Denis Girou. “Présentation de PSTricks”. **in***Cahier GUTenberg*: 16 (**april** 1994), **pages** 21–70.

[6] Michel Goosens **andothers**. *The LaTeX Graphics Companion.* 2 **edition**. Reading, Mass.: Addison-Wesley Publishing Company, 2007.

[7] Alan Hoenig. *TeX Unbound: LaTeX & TeX Strategies, Fonts, Graphics, and More.* London: Oxford University Press, 1998.

[8] Nikolai G. Kollock. *PostScript richtig eingesetzt: vom Konzept zum praktischen Einsatz.* Vaterstetten: IWT, 1989.

[9] Frank Mittelbach **and** Michel Goosens et al. *The LaTeX Companion.* second. Boston: Addison-Wesley Publishing Company, 2004.

[10] Herbert Voß. *`PSTricks` Grafik für TeX und LaTeX.* 7 **edition**. Heidelberg/Berlin: DANTE – Lehmanns, 2016.

[11] Herbert Voß. *`PSTricks` Graphics for LaTeX.* 1 **edition**. Cambridge: UIT, 2011.

[12] Timothy Van Zandt. *`multido.tex` - a loop macro, that supports fixed-point addition.* CTAN, 1997. URL: `/graphics/pstricks/generic/multido.tex`.

[13] Timothy Van Zandt. *PSTricks - `PostScript` macros for generic TeX.* TeX Users Group. 1993. URL: `http://www.tug.org/application/PSTricks` (**urlseen** 21/08/2016).

[14] Timothy Van Zandt **and** Denis Girou. “Inside PSTricks”. **in***TUGboat*: 15 (**september** 1994), **pages** 239–246.

[15] Timothy Van Zandt **and** Herbert Voß. *`pst-plot`: Plotting two dimensional functions and data.* CTAN, 2016. URL: `graphics/pstricks/generic/pst-plot.tex`.

Index

bcurveTension, 5, 7

curvature, 2, 8

dvips, 3

false, 5
File
    pst-bezier.pro, 3
    pst-bezier.sty, 3
    pst-bezier.tex, 3
fillstyle, 8

Keyword
    bcurveTension, 5, 7
    curvature, 2, 8
    fillstyle, 8
    showpoints, 4, 5, 8
    solid, 8

L, 5
l, 5

Macro
    \psbcurve, 2–4, 7, 8
    \psbezier, 2, 3
    \pscurve, 2, 3
    \psRQBCmasse, 10
    \psset, 7
    \usepackage, 3

Package
    pst-bezier, 3
    pstricks, 2, 3
Program
    dvips, 3
\psbcurve, 2–4, 7, 8
\psbezier, 2, 3
\pscurve, 2, 3
\psRQBCmasse, 10
\psset, 7
pst-bezier, 3
pst-bezier.pro, 3
pst-bezier.sty, 3
pst-bezier.tex, 3
pstricks, 2, 3

r, 5

showpoints, 4, 5, 8
solid, 8
spline, 2
Syntax
    L, 5
    l, 5
    r, 5
    T, 7, 8
    t, 6
    tl, 6
    tr, 6
    ts, 7

T, 7, 8
t, 6
tl, 6
tr, 6
ts, 7

\usepackage, 3

Value
    false, 5