

# bezierplot

Linus Romer

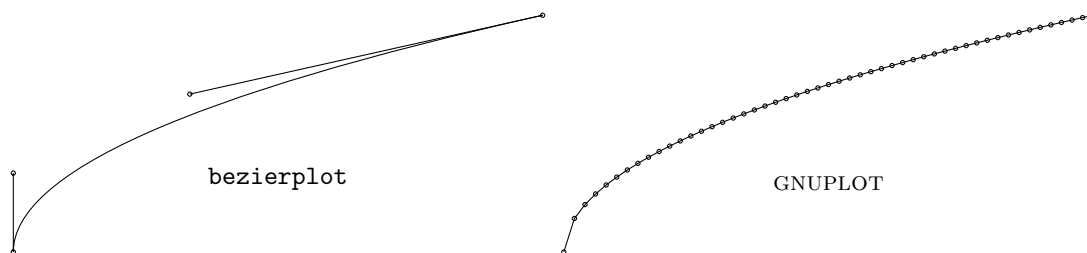
June 2, 2018

## 1 Introduction

**bezierplot** is a Lua program as well as a (Lua) $\LaTeX$  package. This document describes both.

Given a smooth function, **bezierplot** returns a smooth bezier path written in  $\text{TikZ}$  notation (which also matches  $\text{METAPOST}$ ) that approximates the graph of the function. For polynomial functions of degree  $\leq 3$  and inverses of them, the approximation is exact. **bezierplot** finds special graph points such as extreme points and inflection points and reduces the number of used points.

The following example will show a comparison of **GNUPLOT** with **bezierplot** for the function  $y = \sqrt{x}$  for  $0 \leq x \leq 5$ :



**GNUPLOT** used 51 samples (no smoothing) and is still quite inexact at the beginning, whereas **bezierplot** uses 4 points only and is exact!

## 2 Installation

As **bezierplot** is written in Lua, the installation depends whether you are using  $\text{Lua}\LaTeX$  or another  $\LaTeX$  engine.

### 2.1 Installation For $\text{Lua}\LaTeX$

If you have installed **bezierplot** by a package manager, the installation is already complete. The manual installation of **bezierplot** is done in 2 steps:

- copy the files **bezierplot.lua** and **bezierplot.sty** somewhere in your **texmf** tree (e.g. to `~/texmf/tex/lualatex/bezierplot/bezierplot.sty` and `~/texmf/scripts/bezierplot/bezierplot.lua`)
- update the **ls-R** databases by running **mktexlsr**

### 2.2 Additional Installation Steps For Other $\LaTeX$ Engines

You will have to call **bezierplot** as an external program via the option `--shell-escape` (`--write18` for  $\text{MiK}\LaTeX$ ). Therefore, **bezierplot.lua** has to be copied with the name **bezierplot** to a place, where your OS can find it. Under Linux this usually means copying to the directory `/usr/local/bin/`, but for Windows this will probably include more steps

(like adding to the `PATH`). Of course, Lua has to be installed as well. As soon as you can call `bezierplot` from a command line (e.g. by typing `bezierplot "x^2"`), it should also work with other  $\text{\LaTeX}$  engines.

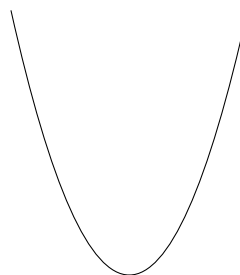
### 3 Loading

The `bezierplot` package is loaded with `\usepackage{bezierplot}`. There are no loading options for the package.

### 4 Usage

A minimal example of  $\text{\LaTeX}$  document could be:

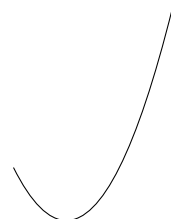
```
\documentclass{article}
\usepackage{tikz,bezierplot}
\begin{document}
\tikz \draw \bezierplot{x^2};
\end{document}
```



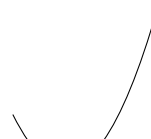
The command `\bezierplot` has 4 optional arguments in the sense of

`\bezierplot[XMIN][XMAX][YMIN][YMAX]{FUNCTION}`

The defaults are  $XMIN = YMIN = -5$  and  $XMAX = YMAX = 5$ .



`\bezierplot[-1][2]{x^2}`



`\bezierplot[-1][2][0.5][3]{x^2}`

You may reverse the graph by making  $XMIN$  bigger than  $XMAX$ . E.g.

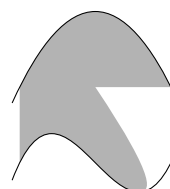
`\bezierplot[-5][5]{0.5*x+1}`

returns  $(-5, -1.5) \dashrightarrow (5, 3.5)$ , whereas

`\bezierplot[5][-5]{0.5*x+1}`

returns the reversed path  $(5, 3.5) \dashrightarrow (-5, -1.5)$ . This is useful, if you want to cycle a path to a closed area:

```
\begin{tikzpicture}
\fill[black!30] \bezierplot[-1][1]{2-x^2}
-- \bezierplot[1][-1]{x^3-x} -- cycle;
\draw \bezierplot[-1.1][1.1]{2-x^2};
\draw \bezierplot[-1.1][1.1]{x^3-x};
\end{tikzpicture}
```



## 4.1 Running Raw bezierplot

Of course, you can run `bezierplot.lua` in a terminal without using L<sup>A</sup>T<sub>E</sub>X, e.g.

```
lua bezierplot.lua "3*x^0.8+2"
```

will return

(0,2) .. controls (0.03,2.282) and (0.268,3.244) .. (1,5)

You can set the window of the graph as follows:

```
lua bezierplot.lua "FUNCTION" XMIN XMAX YMIN YMAX
```

e.g.

```
lua bezierplot.lua "FUNCTION" 0 1 -3 2.5
```

will set  $0 \leq x \leq 1$  and  $-3 \leq y \leq 2.5$ . You may also omit the  $y$ -range, hence

```
lua bezierplot.lua "FUNCTION" 0 1
```

will set  $0 \leq x \leq 1$  and leave the default  $-5 \leq y \leq 5$ .

## 4.2 Notation Of Functions

The function term given to `bezierplot` must contain at most one variable:  $x$ . E.g. `"2.3*(x-1)^2-3"`.

You must not omit `*` operators:

wrong: ~~`2x(x+1)`~~

correct: `2*x*(x+1)`

You have two possibilities to write powers: `"x^2"` and `"x**2"` both mean  $x^2$ .

The following functions and constants are possible:

<code>abs</code>	absolute value (remember: your function should still be smooth)
<code>acos</code>	$\cos^{-1}$ inverse function of cosine in radians
<code>asin</code>	$\sin^{-1}$ inverse function of sine in radians
<code>atan</code>	$\tan^{-1}$ inverse function of tangent in radians
<code>cbrt</code>	cube root $\sqrt[3]{\phantom{x}}$ that works for negative numbers, too
<code>cos</code>	cosine for angles in radians
<code>exp</code>	the exponential function $e^{(\phantom{x})}$
<code>e</code>	the euler constant $e = \exp(1)$
<code>log</code>	the natural logarithm $\log_e(\phantom{x})$
<code>pi</code>	Archimedes constant $\pi \approx 3.14$
<code>sgn</code>	sign function
<code>sin</code>	sine for angles in radians
<code>sqrt</code>	square root $\sqrt{\phantom{x}}$
<code>tan</code>	tangent for angles in radians

## 5 Examples of bezierplot in Comparison with gnuplot

The following graphs are drawn with `bezierplot` (black) and `GNUPLOT` (red). `GNUPLOT` used 1000 samples per example. The functions are given below the pictures (left: `bezierplot`, right: `GNUPLOT`).

