bezierplot

Linus Romer

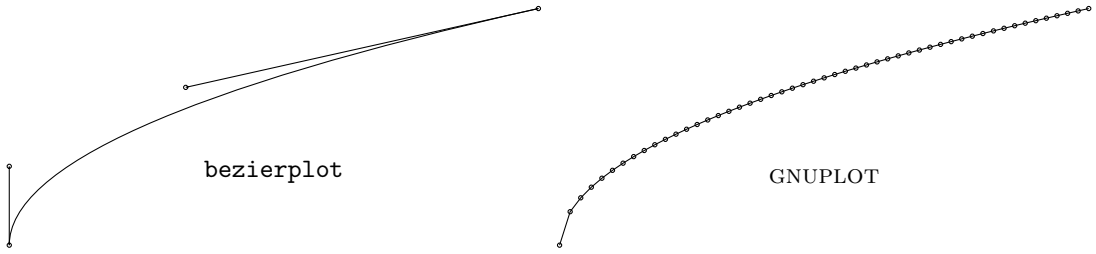
April 11, 2018

1 Introduction

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

Given a smooth function, `bezierplot` returns a smooth bezier path written in TikZ/METAPOST notation that approximates the graph of the function. (For polynomial functions of degree ≤ 3 and inverses of them, the approximation is exact.) It finds special 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 if you are using LuaLaTeX or another LaTeX engine.

2.1 Installation For LuaLaTeX

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

- copy the files `bezierplot.lua` and `bezierplot.sty` somewhere in your `texmf` tree (e.g. `/texmf/`)
- 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 MiKTeX). Therefore, `bezierplot.lua` has to be copied under the name `bezierplot` to a place, where your OS can find it. Under Linux this usually means copying to `/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 LaTeX engines.

3 Loading

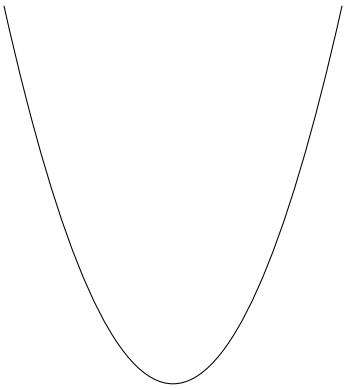
The bezierplot package is loaded with \usepackage{bezierplot}. There are no loading options for the package.

4 Basic Usage

A minimal example of LuaLaTeX document could be:

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

And you will get:

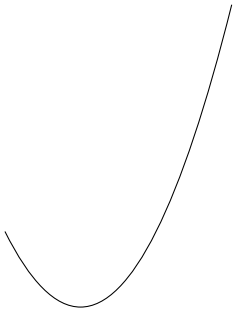


5 Advanced Usage

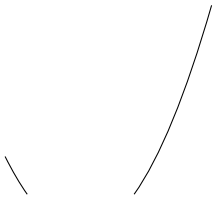
The command \bezierplot has 4 optional arguments in the sense of

```
\bezierplot[XMIN][XMAX][YMIN][YMAX]
```

The defaults are XMIN = YMIN = −5 and XMAX = YMAX = 5.



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



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

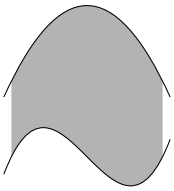
You may reverse the graph by making XMIN bigger than XMAX. E.g.

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

returns `(-5,-1.5) -- (5,3.5)`, whereas

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

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



5.1 Prerequisites

You need to install Lua because `bezierplot` is a Lua script. The purpose of `bezierplot` is the use with Ti*k*Z, so an installation of LaTeX or even LuaTeX and Ti*k*Z is recommended.

5.2 Running `bezierplot`

Download `bezierplot` and run it in a terminal, e.g.

```
lua bezierplot "0.5*x^2-1/(x-2)"
```

or if you make it executable

```
./bezierplot "0.5*x^2-1/(x-2)"
```

or after putting `bezierplot` to an appropriate place even (if you are under Linux I would suggest putting the file in /usr/local/bin):

```
bezierplot "0.5*x^2-1/(x-2)"
```

For the rest of this document, we will assume that you call `bezierplot` as described in the last example.

6 Examples

6.1 Detailed Example

If you want to plot the function $y = 1/(x+1) - 2$ in a window with $-5 \leq x \leq 5$ and $-5 \leq y \leq 5$, you may execute

```
bezierplot "1/(x+1)-2"
```

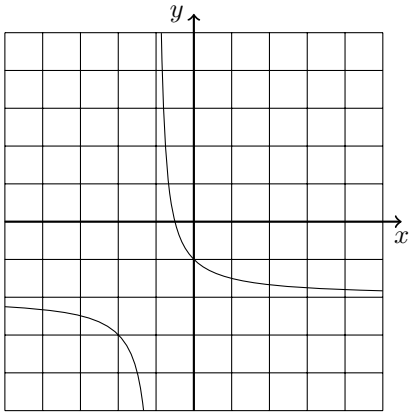
and will get

```
(-5,-2.25) .. controls (-2.12,-2.43) and (-1.59,-2.72) .. (-1.33,-5)
(-0.86,4.99) .. controls (-0.82,3.01) and (-0.75,1.65) .. (-0.64,0.75)
.. controls (-0.53,-0.03) and (-0.38,-0.57) .. (-0.07,-0.92)
.. controls (0.5,-1.59) and (1.7,-1.74) .. (5,-1.83)
```

Include it in a LaTeX file, e.g.

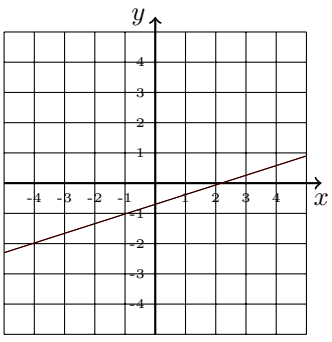
```
\documentclass{article}
\usepackage{tikz}
\begin{document}
\begin{tikzpicture}
\draw (-5,-5) grid (5,5);
\draw[thick,->] (-5,0) -- (5.5,0) node[below]{$x$};
\draw[thick,->] (0,-5) -- (0,5.5) node[left]{$y$};
\draw (-5,-2.25) .. controls (-2.12,-2.43) and (-1.59,-2.72) .. (-1.33,-5)
(-0.86,4.99) .. controls (-0.82,3.01) and (-0.75,1.65) .. (-0.64,0.75)
.. controls (-0.53,-0.03) and (-0.38,-0.57) .. (-0.07,-0.92)
.. controls (0.5,-1.59) and (1.7,-1.74) .. (5,-1.83);
\end{tikzpicture}
\end{document}
```

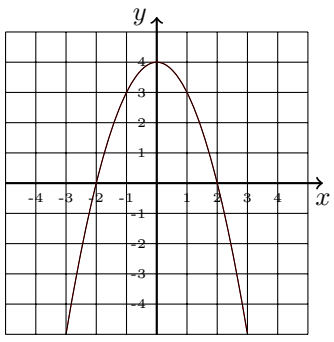
and you will get the following picture:

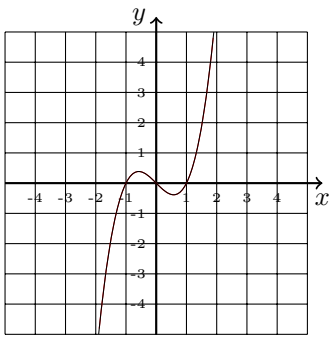


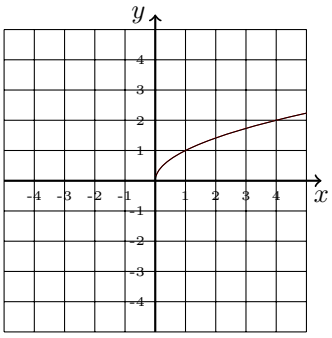
7 Examples of `bezierplot` in Comparison with Gnuplot

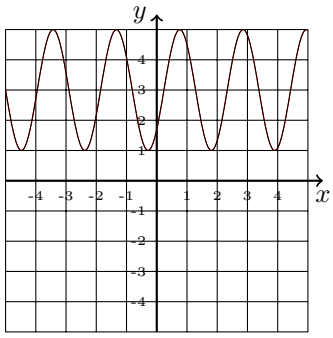
The following graphs are drawn with `bezierplot` (black) and Gnuplot (red). Gnuplot calculated 200 samples per example. The functions are given below the pictures (left: bezierplot, right: Gnuplot).

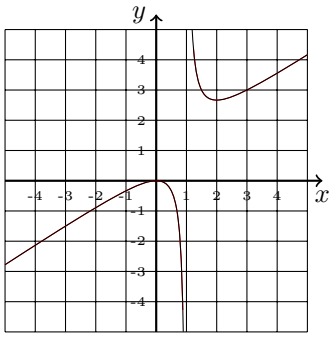


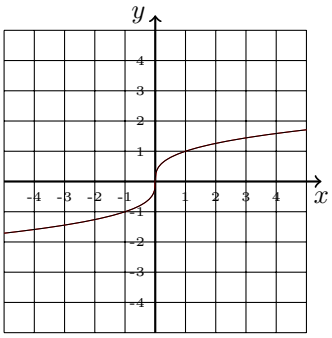


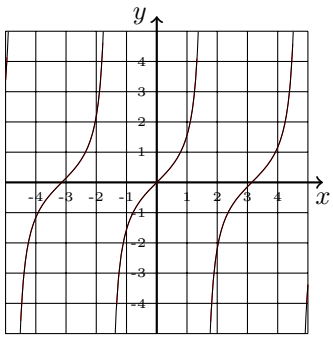


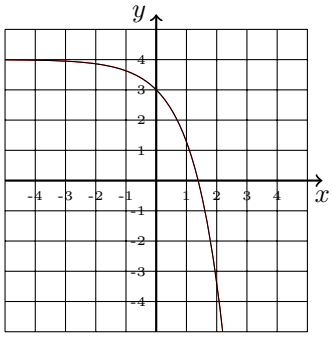


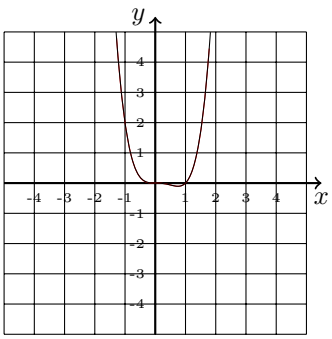


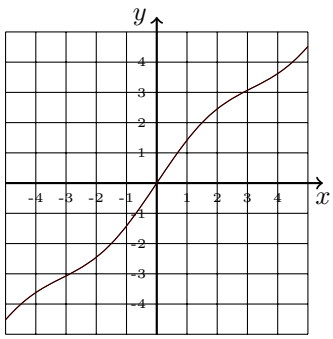


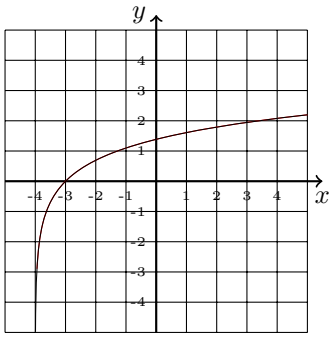












8 Are the Graphs Produced by bezierplot Exact?

The graphs of quadratic and cubic functions and their inverse are exact (up to numeric precision). Sine and cosine functions use the predefined splines from Ti*k*Z (which are very close approximations) if possible. E.g.

```
bezierplot "cos(x)"
```

outputs

```
(-5,0.284) .. controls (-4.909,0.196) and (-4.818,0.105) .. (-4.713,0)
sin (-3.142,-1) cos (-1.571,0) sin (0,1) cos (1.571,0) sin (3.142,-1)
cos (4.713,0) .. controls (4.818,0.105) and (4.909,0.196) .. (5,0.284)
```

9 Options

You can set the window of the graph as follows:

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

e.g.

```
bezierplot "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

```
bezierplot "FUNCTION" 0 1
```

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

10 Daily Use with LaTeX and LuaLaTeX

Supposing your OS finds `bezierplot` automatically (e.g. because it is in `/usr/local/bin`), you can set up your LaTeX file like this:

```
\documentclass{article}
\usepackage{tikz}
\makeatletter\let\evaluatedinput\@@input\makeatother
\providecommand{\bezierplot}[1]{\evaluatedinput|"bezierplot '#1'" }
\begin{document}
\tikz \draw \bezierplot{x^2};
\end{document}
```

If you run LaTeX with enabled shell-escape (option `--shell-escape` for TeXLive, option `--write18` for MiKTeX), you will receive automatically the following picture:

Things get even better with LuaLaTeX, because it can call Lua directly and do not need shell-escape enabled:

```
\documentclass{article}
\usepackage{tikz,xparse}
\directlua{require("bezierplot")}
\DeclareExpandableDocumentCommand{\xbezierplot}{O{-5} O{5} O{-5} O{5} m}{%
\directlua{tex.sprint(bezierplot("#5",#1,#2,#3,#4))}}
\providecommand\bezierplot{\romannumeral`\^^@\xbezierplot}
\begin{document}
\tikz \draw \bezierplot{x^2};
\end{document}
```

The above example also extends the `\bezierplot` command in such way, that you may set the bounds as options: E.g. `\bezierplot[0][1][2]{x^2}` will set $0 \leq x \leq 1$ and $2 \leq y \leq 5$.