1 Introduction

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

Given a smooth function, bezierplot returns a smooth bezier path written in TikZ notation (which also matches METAPOST) that approximates the graph of the function. For polynomial functions of degree ≤ 3 and inverses of them, the approximation is exact (up to numeric precision). 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 \( 0 \leq x \leq 5 \):

![bezierplot vs gnuplot comparison](image)

gnuplot used 51 samples (no smoothing) and is still quite inexact at the beginning, whereas bezierplot uses 4 points only and is exact (up to numeric precision)!

2 Installation

As bezierplot is written in Lua, the installation depends whether you are using LuaLaTEX or another LATHTEX engine.

2.1 Installation For LuaLaTEX

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 LATHTEX Engines

You will have to call bezierplot as an external program via the option \(--shell-escape\)\(^{1}\) for MiKTeX). 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 \texttt{bezierplot} from a command line (e.g. by typing \texttt{bezierplot "x^2"}), it should also work with other \LaTeX engines.

3 Loading

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

4 Usage

A minimal example of Lua\LaTeX document could be:

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

The command \texttt{\bezierplot} has 6 optional arguments in the sense of

\texttt{\bezierplot[XMIN][XMAX][YMIN][YMAX][SAMPLES]{FUNCTION}}

The defaults are \texttt{XMIN = YMIN = -5}, \texttt{XMAX = YMAX = 5} and \texttt{SAMPLES = 0} (this will set as few samples as possible).

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

You may reverse the graph by making \texttt{XMIN} bigger than \texttt{XMAX}. E.g.

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

returns \texttt{(-5,-1.5) -- (5,3.5)}, whereas

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

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

\begin{verbatim}
\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}
\end{verbatim}
4.1 Running Raw bezierplot

Of course, you can run `bezierplot.lua` in a terminal without using \LaTeX, e.g.

```lua
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 and the number of samples as follows:

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

e.g.

```lua
lua bezierplot.lua "FUNCTION" 0 1 -3 2.5 201
```

will set $0 \leq x \leq 1$ and $-3 \leq y \leq 2.5$ and 201 equidistant samples. You may also omit the $y$-range, hence

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

will set $0 \leq x \leq 1$ and leave the default $-5 \leq y \leq 5$. The variables `XMIN`, `XMAX`, `YMIN` and `YMAX` may also be computable expressions like `2*pi+6`:

```lua
lua bezierplot.lua "sin(x)" -pi pi
```

You may use `huge` for $\infty$:

```lua
lua bezierplot "1/x" 0 1 0 huge
```

As `huge` is very huge and `bezierplot` uses recursive calls for nontrivial functions and non-fixed samples, this can last very long:

```lua
lua bezierplot "1/x" -5 5 -huge huge
```

But if you set fixed samples, it will be fast again (as this does not use recursive calls):

```lua
lua bezierplot "1/x" -5 5 -huge huge 100
```

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:

\[
\text{wrong: } 2x(x+1) \quad \text{correct: } 2*x*(x+1)
\]

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

The following functions and constants are possible:

- `abs`: absolute value (remember: your function should still be smooth)
- `acos`: $\cos^{-1}$ inverse function of cosine in radians
- `asin`: $\sin^{-1}$ inverse function of sine in radians
- `atan`: $\tan^{-1}$ inverse function of tangent in radians
- `cbrt`: cube root $\sqrt[3]{\cdot}$ that works for negative numbers, too
- `cos`: cosine for angles in radians
- `exp`: the exponential function $e^{(\cdot)}$
- `huge`: the numerical $\infty$
- `e`: the euler constant $e = \exp(1)$
- `log`: the natural logarithm $\log_e(\cdot)$
- `pi`: Archimedes constant $\pi \approx 3.14$
- `sgn`: sign function
- `sin`: sine for angles in radians
- `sqrt`: square root $\sqrt{\cdot}$
- `tan`: tangent for angles in radians
5 Examples of bezierplot in Comparison with gnuplot

The following graphs are drawn with bezierplot (black) and GNUPLOT (red). You may not recognize the red behind the black unless you zoom in. GNUPLOT used 1000 samples per example. The functions are given below the pictures (left: bezierplot, right: GNUPLOT).