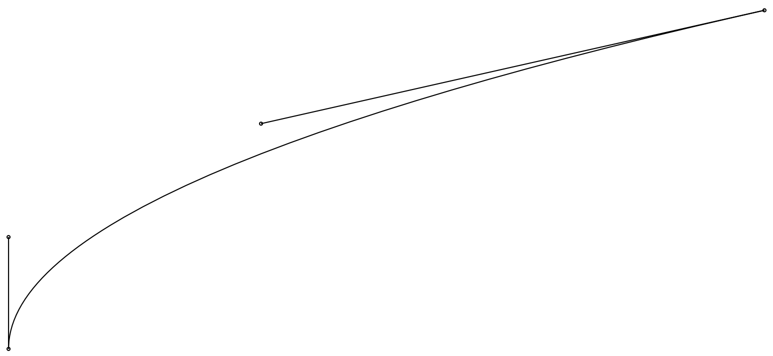
bezierplot

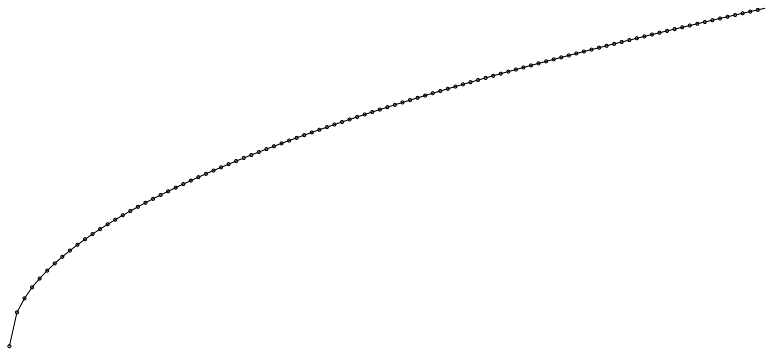
Linus Romer

March 27, 2018

Given a smooth function, `bezierplot` returns a smooth bezier path written in Ti*k*Z. It finds special points such as extreme points and inflection points and reduces the number of used points.



The upper graph of $y = \sqrt{x}$ used `bezierplot`, the lower used the built-in plotting function of Ti*k*Z with 101 samples (no smoothing) and is still quite inexact at the beginning.



1 Getting Started

1.1 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.

1.2 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.

2 Examples

2.1 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"
```

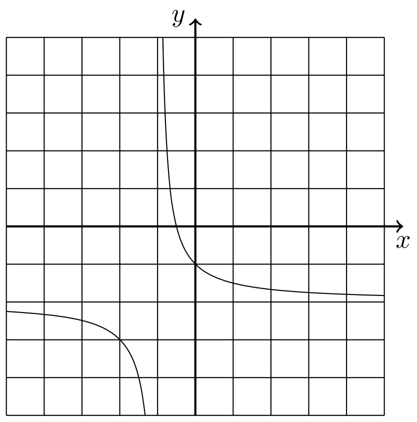
and will get

```
(-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.

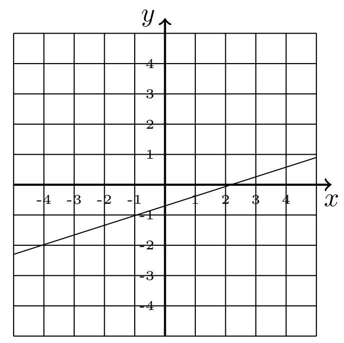
```
\documentclass{article}
\usepackage{tikz}
\begin{document}
\begin{tikzpicture}
\draw (-5,-5) grid (5,5);
\draw[thick,->] (-5,0) -- (5.5,0) node[below]{$x$};
\draw[thick,->] (0,-5) -- (0,5.5) node[left]{$y$};
\draw (-5,-2.25) .. controls (-2.12,-2.43) and (-1.59,-2.72) .. (-1.33,-5)
(-0.86,4.99) .. 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:

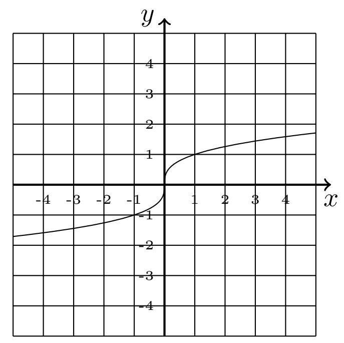


3 Examples of `bezierplot` in Comparison with Gnuplot

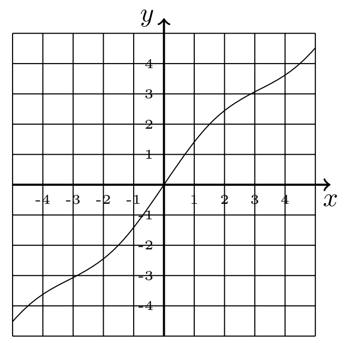
The following graphs are drawn with **`bezierplot`** (black) and Gnuplot (red). Gnuplot calculated 200 samples per example. The functions are given below the pictures (left: bezierplot, right: Gnuplot).



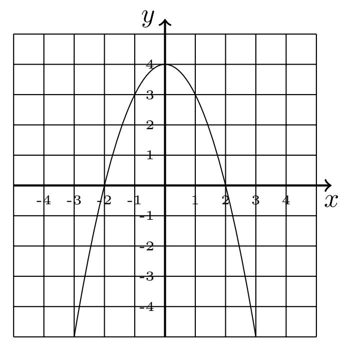
`0.32*x-0.7  | 0.32*x-0.7`



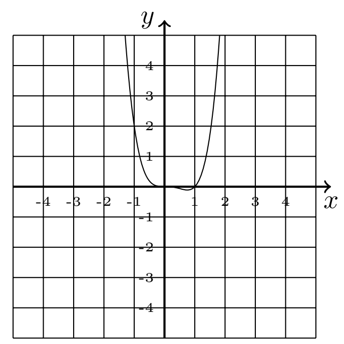
`cbrt(x)  | sgn(x)*abs(x)**(1/3.)`



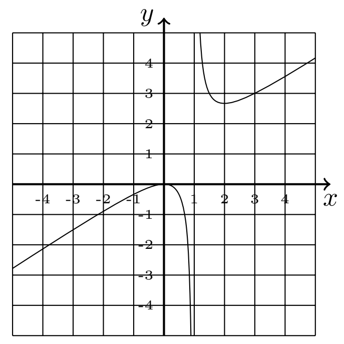
`x+0.5*sin(x)  | x+0.5*sin(x)`



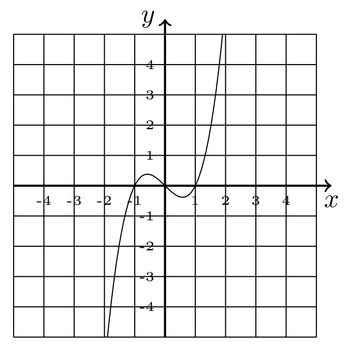
`-x^2+4  | -x**2+4`



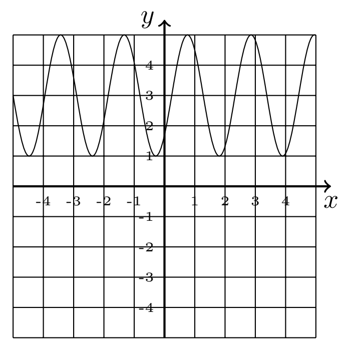
`x^3*(x-1)  | x**3*(x-1)`



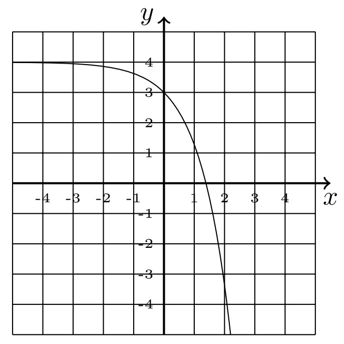
`2*x^2/(3*x-3)  | 2*x**2/(3*x-3)`



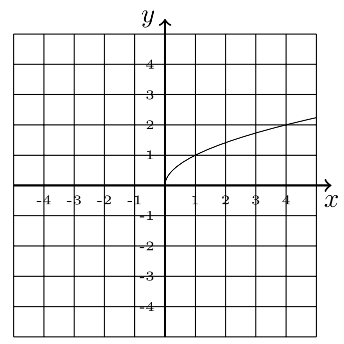
`(x+1)*x*(x-1)  | (x+1)*x*(x-1)`



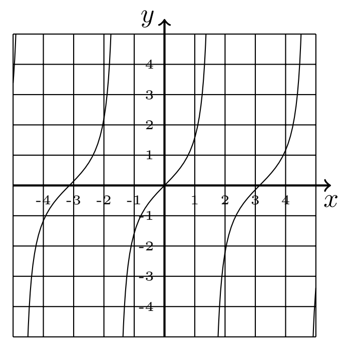
`2*cos(3*x+4)+3  | 2*cos(3*x+4)+3`



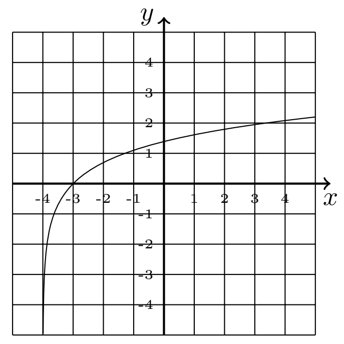
`4-exp(x)  | 4-exp(x)`



`x^0.5  | x**0.5`



`tan(x)  | tan(x)`



`log(x+4)  | log(x+4)`

4 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

```
(-5,0.28) .. controls (-4.91,0.2) and (-4.82,0.11) .. (-4.71,0)
sin (-3.14,-1) cos (-1.57,0) sin (0,1) cos (1.57,0) sin (3.14,-1)
cos (4.71,0) .. controls (4.82,0.11) and (4.91,0.2) .. (5,0.28)
```

5 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
```

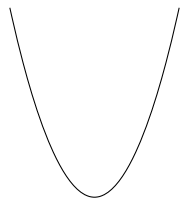
will set $0 \leq x \leq 1$ and leave the default $-5 \leq y \leq 5$.

6 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 $2 \leq y \leq 5$.