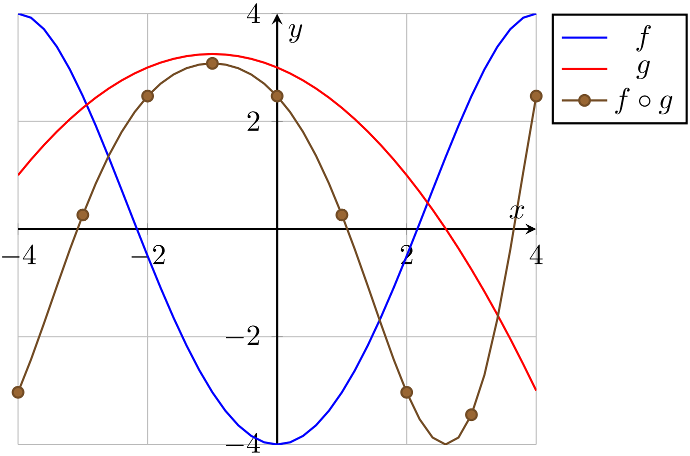
pgfplots generates beautiful simple graphs

Tony Roberts
University of Adelaide
http://orcid.org/0000-0001-8930-1552

May 17, 2018

For research, lecture notes, tutorials, examinations, and online quizzes we often need to simply generate high quality graphs. The LaTeX package pgfplots does a beautiful and flexible job such as the following. This document concisely summarises some useful basics of 2D pgfplots (3D is available but not addressed here).



Contents

1 Basic figure template

Enable with

```
\usepackage{pgfplots}
\pgfplotsset{compat=newest}
```

in the preamble of any regular LaTeX document. I prefer to first draft a graph interactively using the application LaTeXiT. The general format for drawing a figure (often within a `center` environment) is

for a single curve use

```
\begin{tikzpicture}
  \begin{axis}[axis-options]
    \addplot+ formula;
  \end{axis}
\end{tikzpicture}
```

for multiple curves use

```
\begin{tikzpicture}
```

```
  \begin{axis}[axis-options]
    \addplot+[plot-options] formula;
    \addlegendentry{label}
    \addplot+[plot-options] formula;
    \addlegendentry{label}
    ...
  \end{axis}
\end{tikzpicture}
```

2 Graph formulas

function plot To plot a curve where the vertical coordinate is a function of the horizontal coordinate, just give the function formula in terms of `x` within braces. For example

```
\addplot+ {-4+x^2-x^4/32};
```

Trigonometric functions assume degrees, so invoke as `sin(deg(x))` for example, and convert arc-functions as in `atan(x)/deg(1)`.

The "+" in `\addplot+` means that a different line style/colour is used for each successive plot.

parametric plot For a parametric plot give the horizontal and vertical formulas in terms of '`x`' within braces, comma separated, within parentheses. For example, to plot $y = \sqrt{2x-4}$ one could do

```
\addplot+ ({x^2/2+2},{x});
```

given data In place of `formula`, use `coordinates{point-list}` where the point-list has the form `(x1,y1)(x2,y2)...(xn,yn)` for the numerical data point coordinates (no commas between the parentheses). For example, to draw the absolute value function one could

```
\addplot+ coordinates{(-2,2)(0,0)(2,2)}
```

Legend? specify `\addlegendentry{...}` immediately after the curve plot.

Annotation? specifying `\node at (axis cs:x1,y1) {text};` annotates a plot with the text centered at the location `(x1,y1)` in the plot coordinate system.

For example the following draws the figure shown at the start of this document, and using some of the options explained next.

```
\begin{tikzpicture}
\begin{axis}[ xlabel={$x$}, ylabel={$y$}
  ,axis lines=middle
  ,samples=41, grid, thick
  ,domain=-4:4
  ,legend pos=outer north east
  ]
\addplot+[no marks] {-4+x^2-x^4/32};
\addlegendentry{$f$}
\addplot+[no marks] {13/4-(x+1)^2/4};
\addlegendentry{$g$}
\addplot+[mark=*,mark repeat=5]
  {-4+(13/4-(x+1)^2/4)^2-(13/4-(x+1)^2/4)^4/32};
\addlegendentry{$f\circ g$}
\end{axis}
\end{tikzpicture}
```

3 Options

The options for the axis-options and the plot-options are largely the same: it is just that the plot-options override corresponding attributes set in the axis-options.

Multiple options need to be comma separated, and may span many lines. All options are optional, but some are usual.

- `axis lines=middle` pgfplot graphics normally are boxed, but for many purposes we want axes through the origin, so often invoke this.

- `xlabel={$x$}` defines horizontal axis label.
- `ylabel={$y$}` defines vertical axis label; sometimes useful for labelling the plotted function as in `ylabel={$y=\sin x$}`.
- `title={...}` defines a title to go across the top of the plot when necessary.
- `samples=41` The pgfplot default is to use a distressingly few points to approximate a curve; overriding it, to say 41, is common.
- `smooth` Draws a smooth curve between data points (is an alternative to `samples`), especially useful for plots from specified coordinate points.
- `thick` specifies the curves are drawn a bit thicker, which usually seems good to do.
- `grid` for some plots we want a grid drawn.
- `legend pos=...` specifies the position of the legend in a multi-curve figure: can be one of `outer north east` (safely outside the plotted area), `north east`, `south east`, `north west`, `south west`.
- `domain=a:b` usually desirable and specifies the domain $[a, b]$ for the variable `x` in the formula; if not a parametric plot, then this will also be the horizontal extent of the plot.
- `xmin=a, xmax=b, ymin=c, ymax=d` any or all of these specify the horizontal and vertical domains of the plot; any curve or data point outside these ranges are clipped out of the plot; needed sometimes.
- colour? To specify colour just write the corresponding word from `blue, red, brown, green, cyan, magenta, yellow, black, gray, white, darkgray, lightgray, lime, olive, orange, pink, purple, teal, violet`.
- `dashed` plots the curve dashed; there is also `solid`, `dotted`, `dashdotted` and `dashdotdotted`.
- `no marks` the default is to mark every ‘data point’ (even if a formulaic curve); usually omit such marks.

Whereas `only marks` omits the line joining the data points.

- `mark=...` to override the default mark; choose from `*` (discs), `x`, `+`, or more via `\usetikzlibrary{plotmarks}` in the preamble.
- `mark repeat=n` instead of marking every data point, this marks every `n`th data point (starting with the first); sometimes useful with specified number of samples.
- `axis equal image` make the axes of equal scaling, and trim width or height to suit.
- `small, footnotesize, tiny` use one of these to make the figure smaller, or even smaller still, or (as it says) tiny, respectively. You may also want to include `font=\small` or `font=\footnotesize` to correspondingly change the size of any annotations.
- `ybar interval,black,fill=pink` will form a (vertical) bar plot with black rectangles and filled with pink. Similarly for `xbar interval`.
- `xtick={-2,...,8}` will force $x$-axis labels and grids to be drawn at every integer between -2 and 8. Whereas `xtick={a,c,...,b}` puts $x$-axis labels and grids at $a : \delta : b$ where $\delta = c - a$. Analogously for `ytick`.
- `xticklabels={list}` will label each $x$-axis tick with specified information. For example, `xtick={1,3,4}` and `xticklabels={$a$,$x$,$b$}` specifies three ticks at these locations but labels them $a$, $x$ and $b$ respectively. Analogously for `yticklabels`.

4 Extras

- `\pgfplotsset{options}` Sets global options so they do not have to be repeated. For example, make all plots small by `\pgfplotsset{small}`.
- The function `rand` generates a random number for each invocation in a function at each data point; the random numbers are uniform over $[-1, 1]$.

- Inequalities provide the step function: for example, `(x>0.5)` is the function which is zero for $x \leq 0.5$ and one for $x > 0.5$.
- `\node[pin=45:{$e$}] at (axis cs:2.71828,0) {};` Put after an `\addplot ...;` annotates a plot with a pin and a marker at the given location.

  `\node[circle,fill=blue,scale=0.5,pin=135:{$(3,24)$}] at (axis cs` additionally draws a circular marker there as well.
- You can mix colours: for example, `teal!50!white` gives a pale teal.
- Option `opacity=`fraction makes something somewhat transparent; for example, `fill opacity=0.5` makes a fill 50% transparent.
- `\addplot[...] {...} \closedcycle;` is useful for shading regions as it draws end-lines and fills-in down to the horizontal axis.
- `hide y axis` does precisely what it says.
- One can add explanatory text to a legend with

  ```
  \addlegendimage{empty legend}
  \addlegendentry[text width=9em,text depth=]{The quick
      brown fox jumps over the lazy dog.}
  ```

  It is typeset ragged-right.
- Contours? Drawing contours is possible, but currently requires tricky interfacing with external software. Instead, if the contours can be parametrised, then use the `\foreach` command to draw all the curves. For example, to draw six circles centred on the origin one might code, via angle parameter `x` and radius parameter `\r`,

  ```
  \foreach \r in {0.5,1,...,3} {
      \addplot+[no marks,domain=0:360,forget plot]
          ({\r*cos(x)},{\r*sin(x)});
      };
  ```

  The `+` increments the line style, but the `forget plot` says to forget that it used the line style, with the combined effect that all curves are drawn with the same line style.

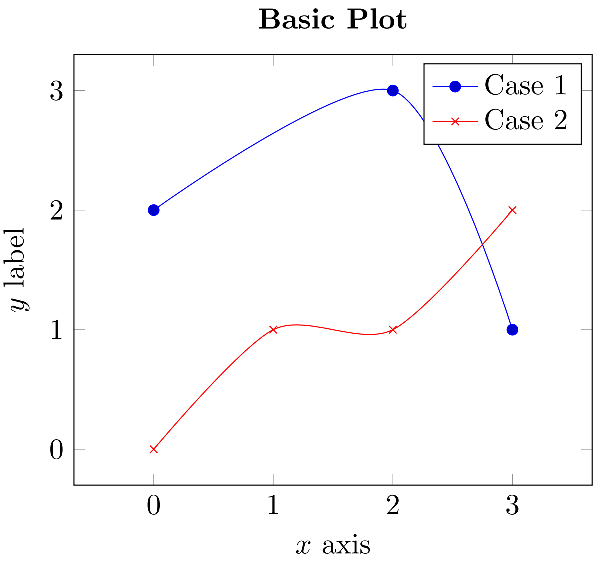
- Drawing the graphs is computationally expensive: if it gets too slow you can get them drawn to a pdf file once and then seamlessly read back in thereafter. In modern systems, the graphs are redrawn automatically when their code is changed (but not in old systems).
  - Invoke this drawing to file by placing in the LaTeX preamble

    ```
    \usepgfplotslibrary{external}
    \tikzexternalize
    ```

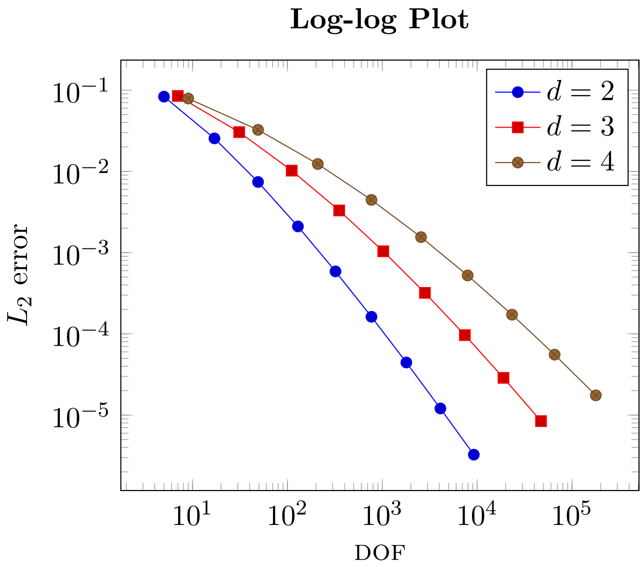
  - `\tikzsetnextfilename{Figs/filename}` It is best to identify precisely what the pdf file is to be called so invoke this command immediately before each and every `\begin{tikzpicture}`. The filename should include a folder, such as `Figs`, because pgfplots generates four files per graph.
  - I like to put the plot source into the file `Figs/filename.tex` (with `\tikzsetnextfilename{Figs/filename}` as its first line), and then invoke in the LaTeX with `\input{Figs/filename}`.
  - Delete a `.md5` file to force the corresponding plot to be redrawn, or invoke `\tikzset{external/force remake}` to force a redraw of all the pgfplots if necessary or when desirable.

5 Further examples

Christian Feuersanger provides many examples, including these.



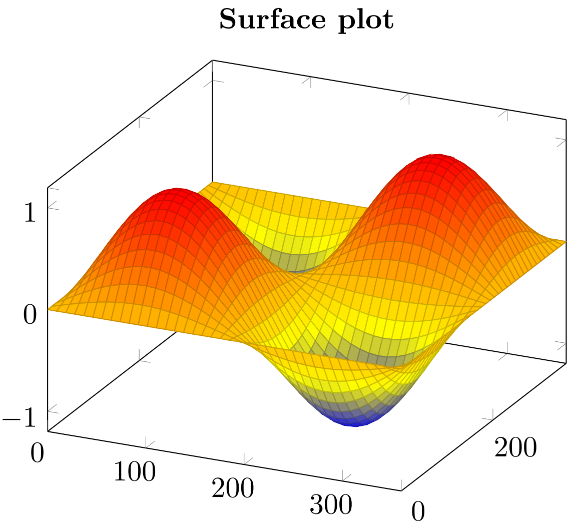
```
\tikzsetnextfilename{Figs/pgfBasicPlot}
\begin{tikzpicture}
    \begin{axis}[axis equal, title={\textbf{Basic Plot}},
        xlabel={$x$ axis}, ylabel={$y$ label}]
    \addplot+[smooth,mark=*] plot coordinates
        { (0,2) (2,3) (3,1) };
    \addlegendentry{Case 1}
    \addplot+[smooth,mark=x] plot coordinates
        { (0,0) (1,1) (2,1) (3,2) };
    \addlegendentry{Case 2}
    \end{axis}
\end{tikzpicture}
```



```
\tikzsetnextfilename{Figs/pgfLoglog}
\begin{tikzpicture}
    \begin{loglogaxis}[title={\textbf{Log-log Plot}},
    xlabel=\textsc{dof}, ylabel={$L_2$ error} ]
      \addplot plot coordinates {
        (5,     8.312e-02)
        (17,    2.547e-02)
        (49,    7.407e-03)
        (129,   2.102e-03)
        (321,   5.874e-04)
        (769,   1.623e-04)
        (1793,  4.442e-05)
        (4097,  1.207e-05)
        (9217,  3.261e-06) };
    \addplot plot coordinates {
        (7,     8.472e-02)
        (31,    3.044e-02)
        (111,   1.022e-02)
        (351,   3.303e-03)
        (1023,  1.039e-03)
        (2815,  3.196e-04)
        (7423,  9.658e-05)
        (18943, 2.873e-05)
        (47103, 8.437e-06) };
    \addplot plot coordinates {
        (9, 7.881e-02)
        (49,    3.243e-02)
        (209,   1.232e-02)
        (769,   4.454e-03)
        (2561,  1.551e-03)
        (7937,  5.236e-04)
        (23297, 1.723e-04)
        (65537, 5.545e-05)
        (178177,    1.751e-05) };
    \legend{$d=2$\\$d=3$\\$d=4$\\}
    \end{loglogaxis}
\end{tikzpicture}
```

6 3D graphics

The simplest plots, such as the one below, are of surfaces expressed as $z = f(x, y)$. Invoke `\addplot3` and express the surface as a function of `x` and `y`.



```
\tikzsetnextfilename{Figs/pgfSurfPlot}
\begin{tikzpicture}
    \begin{axis}[title={\textbf{Surface plot}}]
    \addplot3[surf,domain=0:360,samples=40]
    {sin(x)*sin(y)};
    \end{axis}
\end{tikzpicture}
```

7 matlab2tikz is powerful but requires care

TBA