Manual for Package Pgfplots version 0.9

Christian Feuersänger

January 28, 2008

Abstract

Pgfplots draws high–quality function plots in normal or logarithmic scaling with a user-friendly interface. The user supplies axis labels, legend entries and the plot coordinates for one or more plots and Pgfplots applies axis scaling, computes any logarithms and axis ticks and draws the plots. It is based on Till Tantau's package PGF/Ti*k*Z.

Contents

1 Introduction

This package provides tools to generate plots and labeled axes easily. It draws normal plots, logplots and semi-logplots. Axis ticks, labels, legends (in case of multiple plots) can be added with key-value options. It can cycle through a set of predefined line/marker/color specifications. In summary, its purpose is to simplify the generation of high-quality function plots, especially for use in scientific contexts (logplots).

It is build completely on Ti*k*Z and PGF and may be used as Ti*k*Z library. While Ti*k*Z supports a wide range of plotting utilities, axis generation, labels and legends are usually done with several foreach statements. This is substantially easier with Pgfplots.

2 Installation

2.1 Prerequisites

Pgfplots requires PGF version $\geq$ 1.18 and package `xkeyval`.

2.2 Files in the Pgfplots-Bundle

The Pgfplots package consists of the files

```
latex/pgfplots/pgfplots.sty
doc/latex/pgfplots/manual.bib
doc/latex/pgfplots/manual.tex
doc/latex/pgfplots/manual.pdf
doc/latex/pgfplots/todo.txt
generic/pgfmathlog/pgfmathlog.sty
```

```
generic/liststructure/liststructure.sty
test/liststructuretest/liststructuretest.tex
test/pgfplotstest/pgfplotstest.tex
test/pgfmathlogtest/pgfmathlogtest.tex
shellscripts/pgf2pdf.sh
```

If is used with

```
\usepackage{pgfplots}
```

in your preamble. There are several ways how to teach LaTeX where to find the files. Choose the option which fits your needs best.

2.3 Assigning the `TEXINPUTS` Variable

You can simply install Pgfplots anywhere on your disc, for example into

`/foo/bar/pgfplots.`

Then, you set the `TEXINPUTS` variable to

```
TEXINPUTS=/foo/bar/pgfplots//:
```

The trailing '`:`' tells LaTeX to check the default search paths after `/foo/bar/pgfplots`. The double slash '`//`' tells LaTeX to search all subdirectories.

If the `TEXINPUTS` variable already contains something, you can append the line above to the existing `TEXINPUTS` content.

Please refer to your operating systems manual for how to set environment variables.

2.4 Installation into a local `texmf`-directory

Copy Pgfplots to a local `texmf` directory like `~/texmf` in your home directory. Then, install Pgfplots into the subdirectory `texmf/tex/generic/pgfplots` and run `texhash`.

2.5 Installation into a local TDS compliant `texmf`-directory

A TDS conforming installation will use the same base directory as in the last section, but it requires to merge the contents of '`latex`' into '`texmf/tex/latex`'; the contents of '`generic`' to '`texmf/tex/generic`' and the contents of '`doc`' to '`texmf/doc`'.

Do not forget to run `texhash`.

2.6 If everything else fails...

If LaTeX still doesn't find your files, you can copy all `.sty`-files into your current project's working directory.

Please refer to `http://www.ctan.org/installationadvice/` for more information about package installation.

2.7 Restrictions for DVI-Viewers and `dvipdfm`

PGF is compatible with

- `latex/dvips`,
- `latex/dvipdfm`,
- `pdflatex`,
- ⋮

However, there are some restrictions: I don't know any DVI viewer which is capable of viewing the output of PGF (and therefor Pgfplots as well). After all, DVI has never been designed to draw something different than text and horizontal/vertical lines. You will need to view the postscript file or the pdf-file.

Furthermore, PGF needs to know a *driver* so that the DVI file can be converted to the desired output. Depending on your system, you need the following options:

- `latex/dvips` does not need anything special because `dvips` is the default driver if you invoke `latex`.
- `pdflatex` will also work directly because `pdflatex` will be detected automatically.
- `latex/dvipdfm` requires to use

  ```
  \def\pgfsysdriver{pgfsys-dvipdfm.def}
  %\def\pgfsysdriver{pgfsys-pdftex.def}
  %\def\pgfsysdriver{pgfsys-dvips.def}
  \usepackage{pgfplots}.
  ```

  The uncommented commands could be used to set other drivers explictly.

Please read the corresponding sections in [1, Section 7.2.1 and 7.2.2] if you have further questions. These sections also contain limitations of particular drivers.

3 Drawing axes and plots

3.1 A first plot

Plotting is done using `\begin{axis} ... \addplot ...; \end{axis}`:

```
\begin{tikzpicture}
    \begin{axis}[
        xlabel=Cost,
        ylabel=Error]
    \addplot[color=red,mark=x] plot coordinates {
        (2,-2.8559703)
        (3,-3.5301677)
        (4,-4.3050655)
        (5,-5.1413136)
```

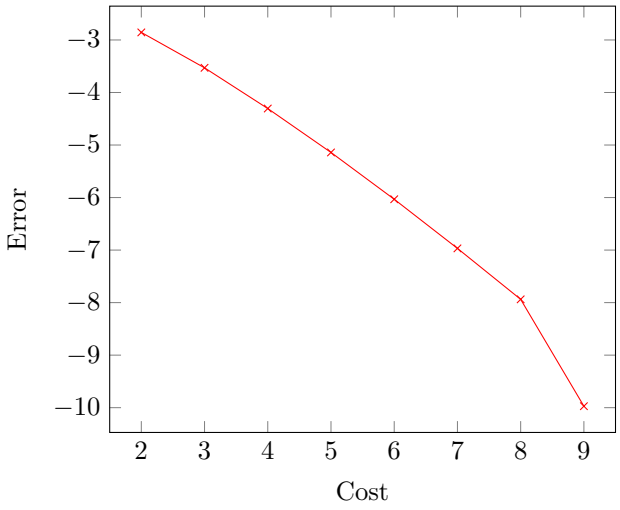


Figure 1: An example for a normal plot. The coordinates are specified using the Ti*k*Z-syntax "`plot coordinates`", optional labels can be provided with the "`xlabel`" and "`ylabel`" arguments.

```
        (6,-6.0322865)
        (7,-6.9675052)
        (8,-7.9377747)
        (9,-9.9717663)
    };
    \end{axis}
\end{tikzpicture}
```

The outcome of this listing is shown in figure 1. The `plot coordinates` command is one of the Ti*k*Z ways to create plots, see [1, Section 16]. All other commands are used to create the axis.

3.2 Two plots in the same axis

Figure 2 shows the result of placing multiple `\addplot`-commands into a single axis:

```
\begin{tikzpicture}
    \begin{axis}[
        xlabel=Cost,
        ylabel=Error]
    \addplot[color=red,mark=x] plot coordinates {
        (2,-2.8559703)
        (3,-3.5301677)
        (4,-4.3050655)
        (5,-5.1413136)
        (6,-6.0322865)
        (7,-6.9675052)
        (8,-7.9377747)
```

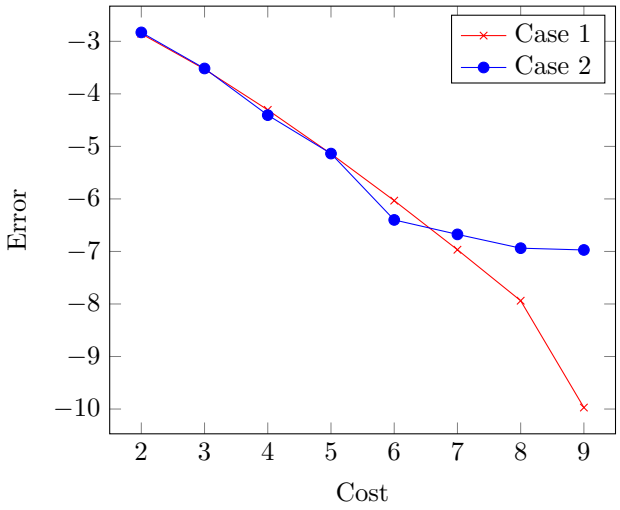


Figure 2: Two plots in the same axis. A legend can be generated using the `\legend`-command.

```
        (9,-9.9717663)
    };
    \addplot[color=blue,mark=*] plot coordinates {
        (2,-2.83)
        (3,-3.5167)
        (4,-4.4050)
        (5,-5.137)
        (6,-6.4)
        (7,-6.6750)
        (8,-6.9377)
        (9,-6.9717)
    };
    \legend{Case 1\\Case 2\\}
    \end{axis}
\end{tikzpicture}
```

3.3 Logarithmic plots

Logarithmic plots show $\log x$ versus $\log y$ (or just one logarithmic axis) as in figure 3. Pgfplots always uses the natural logarithm, i.e. basis $e \approx 2.718$. Now, the axis description also contains minor ticks and the labels are placed at 10^i.

```
\begin{tikzpicture}
\begin{loglogaxis}[xlabel=Cost,ylabel=Gain]
\addplot[color=red,mark=x] plot coordinates {
    (10,100)
    (20,150)
    (40,225)
    (80,340)
```

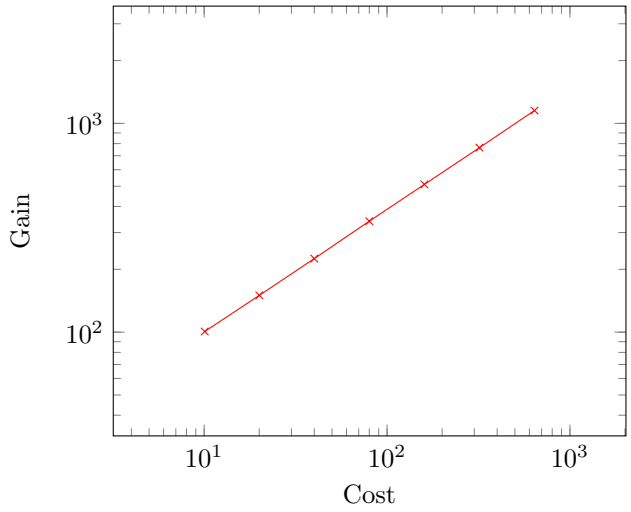


Figure 3: A double–logarithmic plot.

```
    (160,510)
    (320,765)
    (640,1150)
};
\end{loglogaxis}
\end{tikzpicture}
```

A common application is to visualise scientific data. This is often provided in the format $1.42 \cdot 10^4$, usually written as 1.42e+04. An example is shown in the following listing and in figure 4.

```
\begin{tikzpicture}
    \begin{loglogaxis}[
        xlabel=Cost,
        ylabel=Error]
    \addplot[color=red,mark=x] plot coordinates {
        (5,     8.31160034e-02)
        (17,    2.54685628e-02)
        (49,    7.40715288e-03)
        (129,   2.10192154e-03)
        (321,   5.87352989e-04)
        (769,   1.62269942e-04)
        (1793,  4.44248889e-05)
        (4097,  1.20714122e-05)
        (9217,  3.26101452e-06)
    };

    \addplot[color=blue,mark=*] plot coordinates {
        (7,     8.47178381e-02)
        (31,    3.04409349e-02)
        (111,   1.02214539e-02)
```

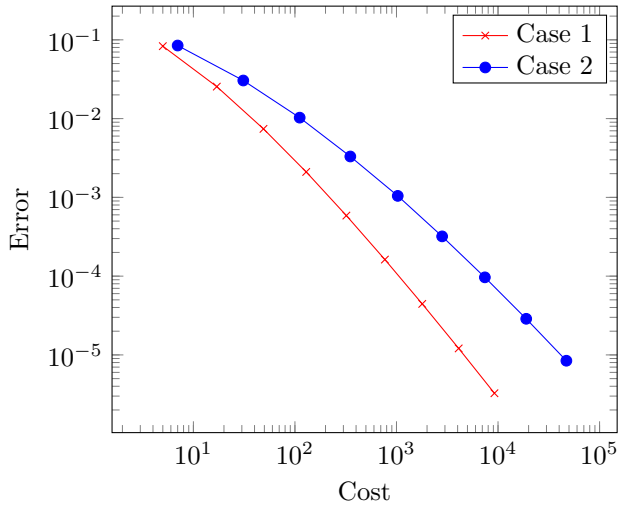


Figure 4: A double–logarithmic plot using scientific notation.

```
        (351,    3.30346265e-03)
        (1023,   1.03886535e-03)
        (2815,   3.19646457e-04)
        (7423,   9.65789766e-05)
        (18943, 2.87339125e-05)
        (47103, 8.43749881e-06)
    };
    \legend{Case 1\\Case 2\\}
    \end{loglogaxis}
\end{tikzpicture}
```

Besided the environment "`loglogaxis`" you can use

- `\begin{axis}...\end{axis}` for normal plots,
- `\begin{semilogxaxis}...\end{semilogxaxis}` for plots which have a normal y axis and a logarithmic x axis,
- `\begin{semilogyaxis}...\end{semilogyaxis}` the same with x and y switched,
- `\begin{loglogaxis}...\end{loglogaxis}` for double–logarithmic plots.

You can also use

```
\begin{axis}[xmode=normal,ymode=log]
...
\end{axis}
```

which is the same as `\begin{semilogyaxis}...\end{semilogyaxis}`. Example:

```
\begin{tikzpicture}
```

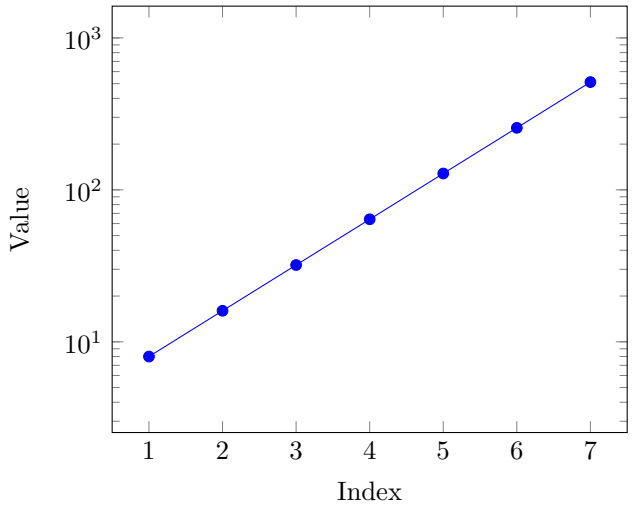


Figure 5: A semi–logarithmic plot.

```
    \begin{semilogyaxis}[xlabel=Index,ylabel=Value]
    \addplot[color=blue,mark=*] plot coordinates {
        (1,8)
        (2,16)
        (3,32)
        (4,64)
        (5,128)
        (6,256)
        (7,512)
    };
    \end{semilogyaxis}
\end{tikzpicture}
```

see figure 5.

3.4 Cycling line styles

You can skip the style arguments for `\addplot[...]` or `\addplot[...]` to determine plot specifications from a predefined list:

```
\begin{tikzpicture}
    \begin{loglogaxis}[
        xlabel={Degrees of freedom},
        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)
```

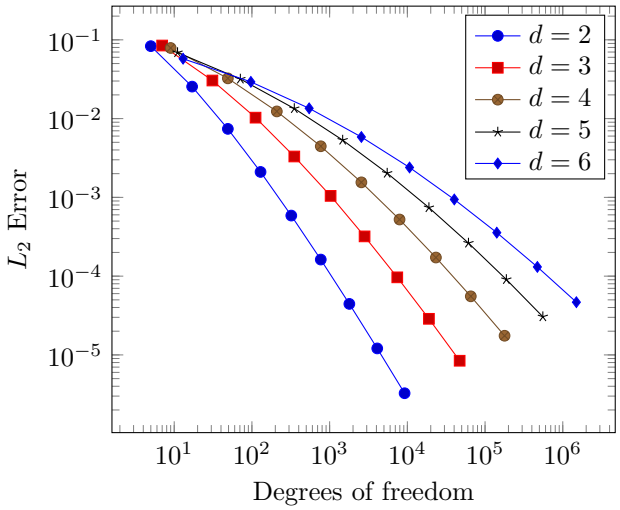


Figure 6: Predefined line/marker combinations.

```
    (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)
};
```

```
    \addplot plot coordinates {
        (11,    6.887e-02)
        (71,    3.177e-02)
        (351,   1.341e-02)
        (1471,  5.334e-03)
        (5503,  2.027e-03)
        (18943, 7.415e-04)
        (61183, 2.628e-04)
        (187903,    9.063e-05)
        (553983,    3.053e-05)
    };

    \addplot plot coordinates {
        (13,    5.755e-02)
        (97,    2.925e-02)
        (545,   1.351e-02)
        (2561,  5.842e-03)
        (10625, 2.397e-03)
        (40193, 9.414e-04)
        (141569,    3.564e-04)
        (471041,    1.308e-04)
        (1496065,   4.670e-05)
    };
    \legend{$d=2$\\$d=3$\\$d=4$\\$d=5$\\$d=6$\\}
    \end{loglogaxis}
\end{tikzpicture}
```

The result is shown in figure 6. You can modify the list, see the reference below.

3.5 Scaling plots

You can use any of the Ti*k*Z options to modify the appearance. For example the effect of the "`scale`" transformation is shown in figures 7 and 8.

You can scale plots either using the `width=5cm` and/or `height=3cm` options or by setting the dimension for each unit coordinate as is shown in figure 9.

More examples can be found in section 5.

4 Reference

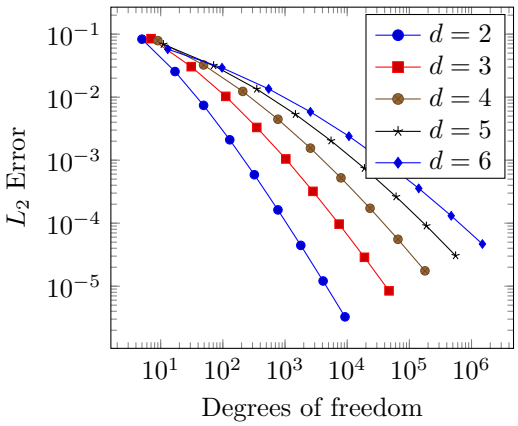
4.1 The axis–environments

There are four axis environments,

1. The axis environment for normal plots,

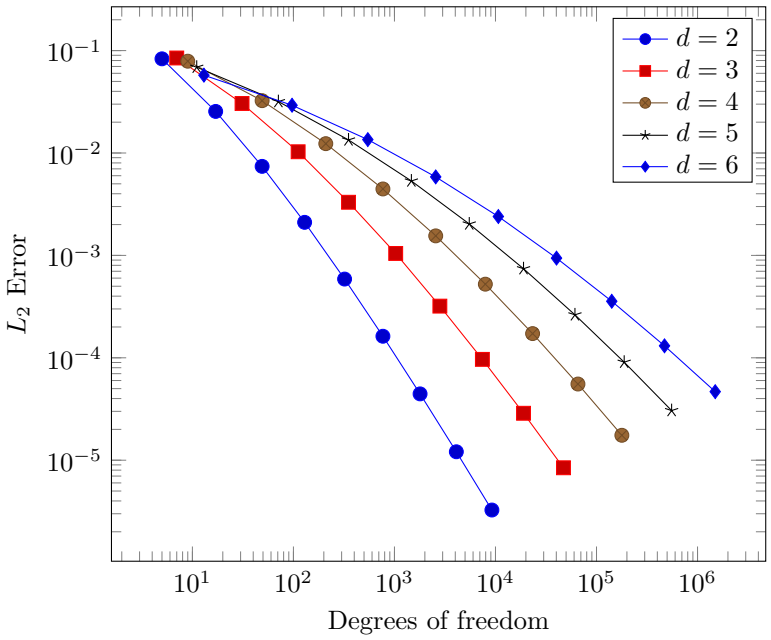
   ```
   \begin{axis}
   ...
   \end{axis}
   ```

2. The axis environment for logarithmic scaling of x and normal scaling of y,



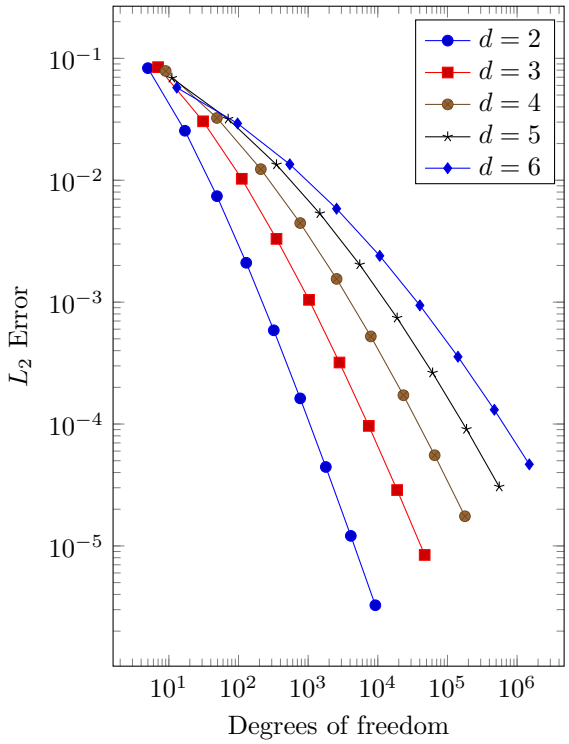

```
\begin{tikzpicture}[scale=0.8]
\begin{loglogaxis}
...
\end{loglogaxis}
\end{tikzpicture}
```

Figure 7: An example of a scaled plot.




```
\begin{tikzpicture}[scale=1.3]
\begin{loglogaxis}
...
\end{loglogaxis}
\end{tikzpicture}
```

Figure 8: The effect of a "`scale`" transformation by 30%.




```
\begin{tikzpicture}
\begin{loglogaxis}[x=0.4cm,y=0.7cm,...]
...
\end{loglogaxis}
\end{tikzpicture}
```

Figure 9: The result of setting the unit vector for x to 0.4cm and for y to 0.7cm.

```
\begin{semilogxaxis}
...
\end{semilogxaxis}
```

3. The axis environment for normal scaling of x and logarithmic scaling of y,

```
\begin{semilogyaxis}
...
\end{semilogyaxis}
```

4. The axis environment for logarithmic scaling of both, x and y axes,

```
\begin{loglogaxis}
...
\end{loglogaxis}
```

They are all equivalent to

```
\begin{axis}[
    xmode=log|normal,
    ymode=log|normal]
...
\end{axis}
```

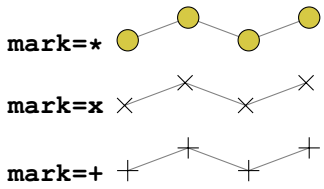
with properly set variables ‘xmode’ and ‘ymode’ (see below).

4.2 Available markers

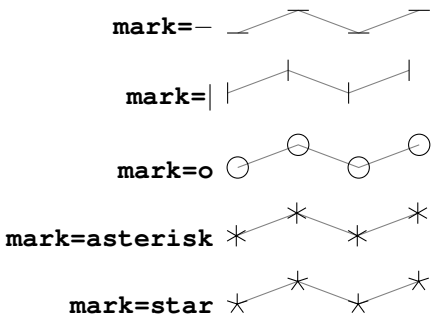
The following options of Ti*k*Z may be useful for plots.

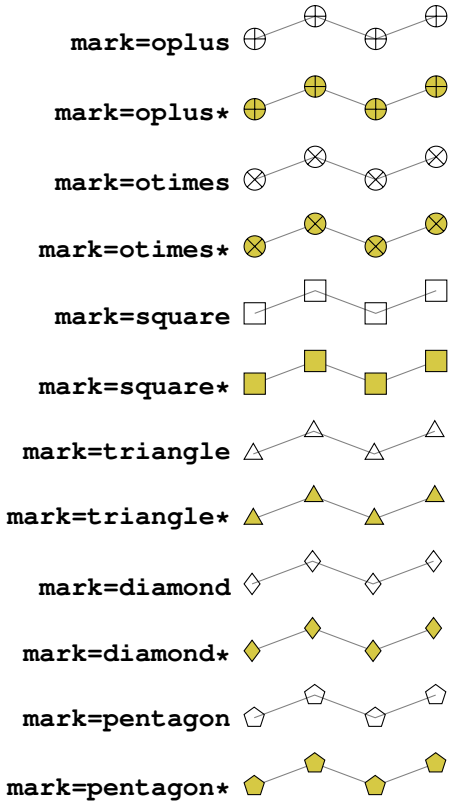
4.2.1 Markers

This list is copied from [1, section 29]:



And with `\usetikzlibrary{plotmarks}`:





All these options have been drawn with the additional options

```
\draw[
    gray,
    thin,
    mark options={scale=2,fill=yellow!80!black,draw=black
    }]
```

4.2.2 Line styles

The following line styles are predefined in Ti*k*Z:

You may need the option `mark options={solid}` to avoid dotted or dashed marker boundaries. The string "`style=`" can be omitted.

4.3 \addplot[OPTIONS] PATH

This is the main plotting command. It is used as

```
\addplot
    plot coordinates {
        (0,0)
        (1,1)
    };
```

or

```
\addplot[color=blue,mark=*]
    plot coordinates {
        (0,0)
        (1,1)
    };
```

The first syntax chooses the next unused plot specification of the list `\autoplotspeclist` (see below) and the second syntax specifies a plot specification explicitly. This specification will be used inside of the legend (if any).
Some more details:

- `\addplot` is a shortcut for `\draw[OPTIONS]`, that means anything up to the next semicolon is part of a PGF-path. The semicolon is required.
- The `OPTIONS` are remembered for the legend.
- See subsection 4.2 for a list of available markers and line styles.
- You can modify `OPTIONS` with

  ```
  \tikzstyle{every axis plot}=[...]
  ```

- For log plots, Pgfplots will compute the natural logarithm $\log(\cdot)$ numerically. This works with normal fixed point numbers or in scientific notation. For example, the following numbers are valid input to `\addplot`.

  ```
  \begin{loglogaxis}
  \addplot plot coordinate {
      (769,   1.6227e-04)
      (1793,  4.4425e-05)
      (4097,  1.2071e-05)
      (9217,  3.2610e-06)
      (1e6,   0.00003341)
      (2.3e7, 0.00131415)
  ;
  \end{loglogaxis}
  ```

  You can represent arbirtrarily small or very large numbers as long as its logarithm can be represented as a TEX-length (up to about 16384). Of course, any coordinate $x \leq 0$ is not possible since the logarithm of a non-positive number is not defined. Such coordinates will be skipped automatically.

- As a consequence of the coordinate parsing routines, you can't use the mathematical expression parsing method of PGF as coordinates.
- If you did not specify axis limits for x and y manually, `\addplot` will compute them automatically.

  The automatic computation of axis limits works as follows:

  1. In case of `\addplot plot coordinates {...};`, every coordinate will be checked. Any other Ti*k*Z-plot mode is not (yet) supported (had no time yet).
  2. Since more than one `\addplot` command may be used inside an `\begin{axis}...\end{axis}`, all drawing commands will be postponed until `\end{axis}`.

     See the `\axispath`-command for more information about automatic axis limits and the postponed drawing.

4.4 \axispath...;

This command allows to draw custom content into an axis. The argument to `\axispath` may be any Ti*k*Z-command like

```
\axispath\node at (12.14368,-9.30872) {};
```

or

```
\axispath\draw (12.14368,-9.30872) -- (0,1);
```

A useful example is presented in section 5.5 where annotations are placed near some data points.

The `\axispath` command is necessary to communicate drawing commands to Pgfplots. If Pgfplots needs to determine the x and/or y limits automatically, any plotting commands (including those with `\axispath`) will be postponed until `\end{axis}`.

A missing `\axispath` may corrupt your graphics because the clipping area may not yet be known.

4.5 \addlegendentry{name}

Adds a single legend entry to the legend list. Example:

```
\begin{axis}
\addplot[smooth,mark=*,blue] plot coordinates {
    (0,2)
    (2,3)
    (3,1)
};
\addlegendentry{Case 1}

\addplot[smooth,color=red,mark=x]
    plot coordinates {
        (0,0)
        (1,1)
```

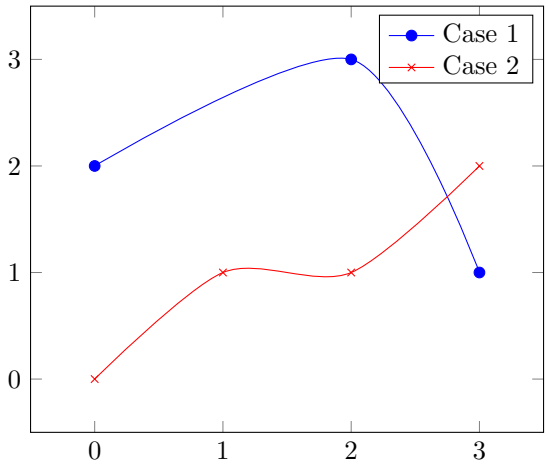


Figure 10: An example for the `\addlegendentry` command

```
        (2,1)
        (3,2)
    };
\addlegendentry{Case 2}
\end{axis}
```

The outcome of this listing is shown in figure 10. It does not matter where `\addlegendentry` commands are placed, only the sequence matters. You will need one `\addlegendentry` for every `\addplot` command.

You may also consider the command `\legend`. It also accepts several formatting options.

4.6 `\legend[OPTIONS]{LIST}`

You can use

```
\begin{tikzpicture}
\begin{axis}
...
\legend{$d=2$\\$d=3$\\$d=4$\\$d=5$\\$d=6$\\}
...
\end{axis}
\end{tikzpicture}
```

to assign a complete legend. The `\legend` command takes the following arguments:

```
\legend[OPTIONS]{TEXT1\\TEXT2\\TEXT3\\...\\}
```

where each legend element needs to be terminated by `\\`. If `LIST` is empty, the current legend will not be touched (see `\addlegendentry`). The short marker/line combination shown in legends is acquired from the argument to `\addplot[...]`.

4.6.1 Legend appearance

The style "every axis legend" determines the legend's position and outer appearance:

```
\tikzstyle{every axis legend}+=[at={(0,0)}]
```

will draw it at the lower left corner of the axis while

```
\tikzstyle{every axis legend}+=[at={(1,1)}]
```

means the upper right corner.

The legend is a node, so you can use any Ti*k*Z option which affects nodes (see [1, section 13]). The node's option "text width" is set automatically, depending on the `\legend[textwidth=...]` option.

Examples:

- `\tikzstyle{every axis legend}+=[at={(1.02,1)},anchor=north west]`

  draws the legend OUTSIDE TOP RIGHT.

- `\tikzstyle{every axis legend}+=[at={(1,0.5)},anchor=east,outer sep=0.5cm]`

  draws the legend INSIDE MIDDLE RIGHT, separated by 0.5cm from the axis.

The default is

```
\tikzstyle{every axis legend}=[%
        anchor=north east,%
        shape=rectangle,%
        fill=white,%
        draw=black,
        at={(0.98,0.98)}
        ]
```

4.6.2 Legends Options

textwidth=auto|none|DIMENSION

Assigns a width to the legend. Choices are

auto Determines the width automatically;

none does not perform any line breaking;

DIMENSION uses a fixed width like `2cm` for the (total) legend (including the small marker/line combination).

Default is "auto".

font=COMMANDS

Changes the legend font. Examples:

- `font=\Huge`
- `font={\Huge\bfseries}`

Please note that the Ti*k*Z option "`font`" will also be used, so you can also say

```
\tikzstyle{every axis legend}+=[font=\Huge]
```

to get the same effect. The difference is that I didn't figure out how to determine the legend's width without a legend option.

4.7 \autoplotspeclist

Allows to specify a list of plot specifications which will be used for each `\addplot`-command without explicit plot specification.

There are several possiblities to change it:

1. Use one of the predefined lists,

   ```
   \listcopy\coloredplotspeclist\to\autoplotspeclist%
   \listcopy\blackwhiteplotspeclist\to\autoplotspeclist%
   ```

2. Define your own one,

   ```
   \listnew{\autoplotspeclist}{%
       blue,mark=*\\%
       red,mark=square\\%
       dashed,mark=o\\%
       loosely dotted,mark=+\\%
       brown!60!black,mark options={fill=brown!40},mark=
       otimes*\\%
   }
   ```

   (This example list requires `\usetikzlibrary{plotmarks}`).

Please note that the '`\\`' is required to separate each element. Please note that comment characters '%' are necessary if the elements are on different lines.

4.8 \logtologten{ARG}

Expands to the result of `ARG`/ log(10).

4.9 \logten

Expands to the constant log(10). Useful for logplots because $\log(10^i) = i \log(10)$.

4.10 \axispreset{key=value,key=value}

Allows to define default options for any axis. For example,

```
...
\axispreset{width=\textwidth}%
...
\begin{tikzpicture}
\begin{axis}
...
\end{axis}
\end{tikzpicture}
```

will produce a width of **\textwidth** for any following axis. You can preset any of the `axis`-options described below.

4.11 \legendpreset{key=value,key=value}

Allows to define default options for any legend (See `\axispreset`, it works in the same way.).

4.12 Axis options

There are several required and even more optional arguments to modify axes. They are used like

```
\begin{tikzpicture}
\begin{axis}[key=value,key2=value2]
...
\end{axis}
\end{tikzpicture}
```

The overall appeareance can be changed with

```
\tikzstyle{every axis}=[line width=1pt]
```

for example.

xmin=COORD, xmax=COORD, ymin=COORD, ymax=COORD

These options denote the axis limits, i.e. the lower left and the upper right corner. Some remarks

- The axis limits determine the plotted range. Everything else will be clipped away.
- The width of every unit x-coordinate will be scaled such that the plot has width `\axisdefaultwidth` (including the tick- and axis labels). The height of every unit y-coordinate will be scaled such that the plot has height `\axisdefaultheight`.

  You can override this default behavior with the options `width=DIMEN`, `height=DIMEN`, `x=DIMEN` and `y=DIMEN`, see below.

- If one of `xmin` or `xmax` is missing, the x-interval will be determined automatically (see `\addplot`). The same holds true if one of `ymin` or `ymax` is missing: in this case, the y-interval will be determined automatically.
- The option `enlargelimits` will automatically increase the plotted range.

`xlabel=TEXT`, `ylabel=TEXT`

Changes the axis labels to '`TEXT`' which is any LaTeX text. Use '`{TEXT}`' if you need grouping.

Labels are Ti*k*Z-Nodes which are placed with

```
\node
    [style=every axis label,
    style=every axis x label]
\node
    [style=every axis label,
    style=every axis y label]
```

so you can reconfigure their position and appearance. As for legends, the coordinate `(0,0)` denotes the lower left axis corner and `(1,1)` the upper right.

The default styles are

```
\tikzstyle{every axis label}=[]
\tikzstyle{every axis x label}=[
    at={(0.5,0)},
    below,
    yshift=-15pt]
\tikzstyle{every axis y label}=[
    at={(0,0.5)},
    xshift=-35pt,
    rotate=90]
```

`xmode=normal`|`log`, `ymode=normal`|`log`

Allows to choose between normal plots or logplots for each x, y-combination. Default is `xmode=normal`, `ymode=normal`.

`xtick=`{`LIST`}, `ytick=`{`LIST`}

Assigns a list of *Positions* where ticks shall be placed. The argument `LIST` will be used inside of a `\foreach \x in {LIST}` statement, and `LIST` contains one of the following formats:

- `{0,1,2,5,8}` (a series of coordinates),
- `{0,...,5}` (the same as `{0,1,2,3,4,5}`),
- `{0,2,...,10}` (the same as `{0,2,4,6,8,10}`),
- `{9,...,3.5}` (the same as `{9, 8, 7, 6, 5, 4}`),
- See [1, Section 34] for a more detailed definition of the options.

For logplots, Pgfplots will apply $\log(\cdot)$ to each element in '`LIST`'.

Attention: You can't use the '`...`' syntax if the elements are too large for TEX! For example, '`xtick=1.5e5,2e7,3e8`' will work (because the elements are interpreted as strings, not as numbers), but '`xtick=1.5,3e5,...,1e10`' will fail because it involves real number arithmetics beyond TEX's capacities.

The default choice for ticks in a normal plot is to place them at each integer. The default for logplots it to place ticks at each 10^i in the axis' range.

The tick appearance can be (re-)configured with

```
\tikzstyle{every tick}=[very thin,gray]
\tikzstyle{every minor tick}=[]
```

This style commands can be used at any time. The tick line width can be configured with the `axis`-options '`tickwidth`' and '`subtickwidth`'.

xtickten={LIST}, ytickten={LIST}

These options allow to place ticks at selected positions $10^k, k \in$ `LIST`. They are only used for logplots. The syntax for '`LIST`' is the same as above for '`xtick=LIST`' or '`ytick=LIST`'.

xticklabel={TEXT}, yticklabel={TEXT}

Allows to change the *text* assigned to each tick position (see options `xtick` and `ytick`). The argument '`TEXT`' can be any LATEX-text. The following commands are valid inside of `TEXT`:

\tick The current element of option `xtick` (or `ytick`).

\ticknum The current tick number, starting with 0 (a counter).

The default argument is

- `xticklabel={$\tick$}` for normal plots and
- `xticklabel={$10^{\logtologten\tick}$}` for logplots

(the same holds for `yticklabel`). You reconfigure the defaults with

```
\renewcommand{\axisdefaultticklabel}[0]{%
    $\tick$%
}
\renewcommand{\axisdefaultticklabellog}[0]{%
    $10^{\logtologten\tick}$%
}
```

You can change the appearance of tick labels with

```
\tikzstyle{every tick label}=[]
```

and/or

```
\tikzstyle{every x tick label}=[]
```

and

```
\tikzstyle{every y tick label}=[]
```

tickpos=left|right|both

Allows to choose where to place the small tick lines. Default is "`both`". This setting applies to both x and y axis where "left" and "right" mean "bottom" and "top" for y.

xminorticks=true|false, yminorticks=true|false

Enables/disables the small minor tick lines. They are only used in log plots. Please note that minor ticks are automatically disabled if `xtick` is not a uniform range[1]. Default is `true`.

tickwidth=DIMEN, subtickwidth=DIMEN

Allows to configure the width of the small tick lines. Sub ticks are only displayed for logarithmic plots.

enlargelimits

Enlarges the axis size somewhat.

You can set `xmin`, `xmax` and `ymin`, `ymax` to the minimum/maximum values of your data and `enlargelimits` will enlarge the canvas such that the axis doesn't touch the plots.

Default is `true` if axis limits are determined automatically and `false` if you provide them manually.

disablelogfilter

Disables numerical evaluation of $\log(x)$ in LaTeX. If you specify this option, any plot coordinates and tick positions must be provided as $\log(x)$ instead of x. This may be faster and – possibly – more accurate than the numerical log. The current implementation of $\log(x)$ normalizes x to $m \cdot 10^e$ and computes

$$\log(x) = \log(m) + e \log(10)$$

where $y = \log(m)$ is computed with a newton method applied to $\exp(y) - m$. The normalization involves string parsing without TeX-registers. You can savely evaluate $\log(1 \cdot 10^{-7})$ although TeX-registers would produce an underflow for such small numbers.

The exponential function is implemented in pgf using a truncated Euler McLaurin–series.

The current implementation of $\log(\cdot)$ is done by the macro `\pgfmathlog` which is provided by package `pgfmathlog.sty` (contained in the Pgfplots-bundle).

width=DIMEN, height=DIMEN

The axis is always scaled such that its dimension is

`(\axisdefaultwidth,\axisdefaultheight)`.

[1] A uniform list means the difference between all elements is 1 for normal plots or, for logplots, $\log(10)$.

The options `width=DIMEN` and `height=DIMEN` override the default scaling. If just one of them is specified, the other one is scaled such that the aspect ratio stays the same.
Remarks:

- The scaling only affects the width of one unit in x-direction or the height for one unit in y-direction. Axis labels and tick labels won't be resized.
- You can use the Ti*k*Z-`scale=NUMBER` option,

```
\begin{tikzpicture}[scale=2]
\begin{axis}
...
\end{axis}
\end{tikzpicture}
```

  to scale the complete picture.
- The Ti*k*Z-options `x` and `y` which set the unit dimensions in x and y directions can be specified as arguments to `\begin{axis}[x=1.5cm,y=2cm]` if needed (see below). These settings override the `width` and `height` options.
- Use

```
\renewcommand{\axisdefaultwidth}{3cm}
\renewcommand{\axisdefaultheight}{6cm}
\begin{axis}
...
\end{axis}
```

  to replace the default dimension.
- Please note that up to the writing of this manual, Pgfplots only estimates the size needed for axis- and tick labels. It does not include legends which have been placed outside of the axis[2]. This may be fixed in future versions.

  Use the `x=DIMEN` and `y=DIMEN` options if the scaling happens to be wrong.

x=DIMEN, y=DIMEN

Use

```
\begin{tikzpicture}
\begin{axis}[x=1.5cm,...]
...
\end{axis}
\end{tikzpicture}
```

to set the unit size for one x-coordinate to 1.5cm. The same is possible for the y-coordinate.

Setting x explicitly overrides the `width` option. Setting y explicitly overrides the `height` option.

[2] I.e. the '`width`' option will not work as expected, but the bounding box is still ok.

Please note that it is *not* possible to specify `x` as argument to `tikzpicture`. The option

```
\begin{tikzpicture}[x=1.5cm]
\begin{axis}
    ...
\end{axis}
\end{tikzpicture}
```

won't have any effect because an axis is always scale to the final size `(\axisdefaultwidth,\axisdefaultheight)` (see the `width` option).

xfilter=CMD, yfilter=CMD

These options allow coordinate filtering. A coordinate filter maps an input coordinate to an output coordinate (or discards it completely).

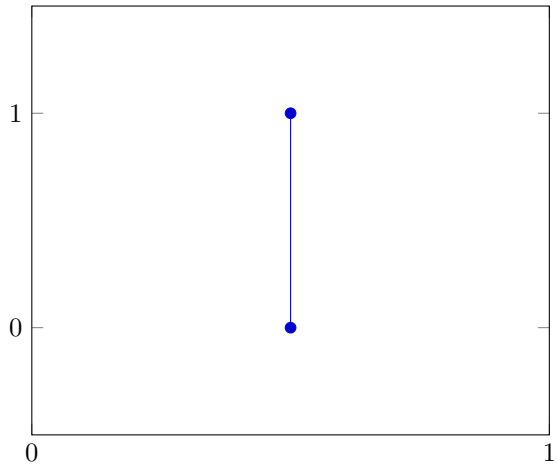
Coordinate filters are useful in automatic processing system, where Pgfplots is used to display automatically generated plots. You may not want to filter your coordinates by hand, so these options provide a tool to do this automatically.

Be warned: '`xfilter`' and '`yfilter`' are advanced options and may require advanced knowledge about TEX.

They are used as in the following example. The code

```
\def\myOwnXfilter#1\to#2{%
    \def#2{0.5}%
}%
\begin{tikzpicture}
\begin{axis}[xfilter={\myOwnXfilter}]
\addplot plot coordinates {
    (4,0)
    (6,1)
};
\end{axis}
\end{tikzpicture}
```

will result in



because all x-coordinates are replaced by 0.5.

The Argument 'CMD' is the name of a TEX-macro which takes exactly two arguments which are separated by the string '\to'. Such a macro is defined as

```
\def\exampleFilter#1\to#2{%
    \def#2{#1}%
}%
```

This example uses the TEX-command **\def** to define variables and commands. The arguments are used as follows:

- Pgfplots invokes the filter with argument #1 set to the input coordinate. For x-filters, this is the x-coordinate as it is specified to \addplot, for y-filters it is the y-coordinate.
- If the corresponding axis is logarithmic, #1 is the *logarithm* of the coordinate as a real number, for example #1=4.2341.
- Argument #2 is the name of a TEX command. The filter should assign this command. The first filter above assigned the constant 0.5 and the second filter did not filter anything because it is the identity.

  The replacement text of #2 is expected to be *either* empty *or* a real number (without any length-suffix like 'cm' or 'pt'). If it is empty, the coordinate won't be drawn at all, it will be thrown away.

4.13 execute at begin plot=COMMANDS

This axis option allows to invoke 'COMMANDS' at the beginning of each \addplot command. The argument 'COMMANDS' can be any LATEX content.

You may use this in conjunction with xfilter=... to reset any counters or whatever. An example would be to change every 4th coordinate.

4.14 execute at end plot=COMMANDS

This axis option allows to invoke 'COMMANDS' after each \addplot command. The argument 'COMMANDS' can be any LATEX content.

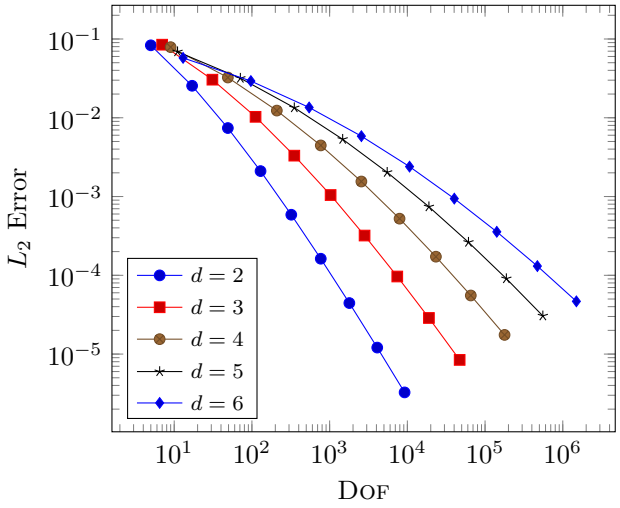
5 More examples

This section contains a catalogue of different Pgfplots features by example.

5.1 Legend position

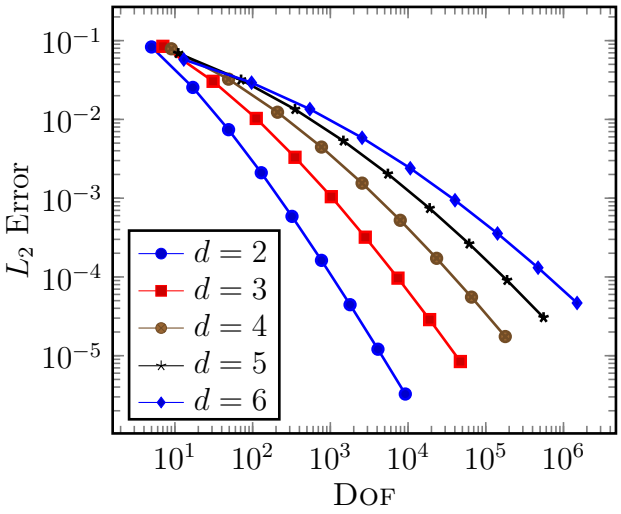
```
\tikzstyle{every axis legend}+=
    [at={(0.03,0.03)},anchor=south west]%
\begin{tikzpicture}
    \begin{loglogaxis}[
        xlabel={\textsc{Dof}},
        ylabel={$L_2$ Error}
    ]
    \addplot plot coordinates {
```

```
        (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)
    };

    ....

    \legend{$d=2$\\$d=3$\\$d=4$\\$d=5$\\$d=6$\\}
    \end{loglogaxis}
\end{tikzpicture}
```



5.2 Font size and line width

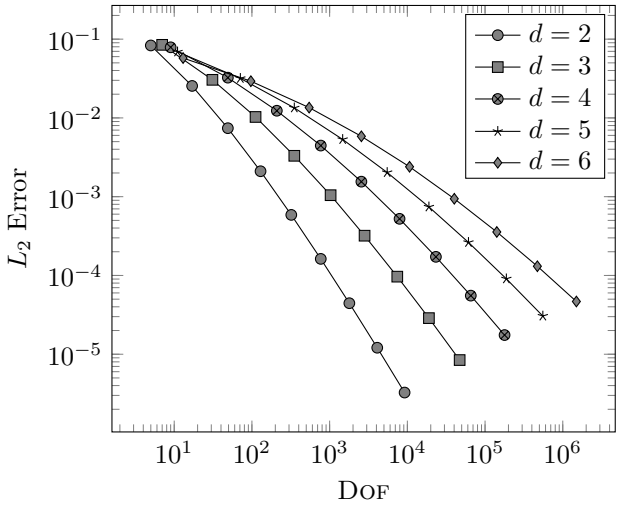
```
\tikzstyle{every axis legend}+=%
    [at={(0.03,0.03)},anchor=south west]%
\tikzstyle{every tick}+=[line width=0.6pt]%
\begin{tikzpicture}[font=\large,line width=1pt]
    \begin{loglogaxis}[
        xlabel={\textsc{Dof}},
        ylabel={$L_2$ Error}
    ]
    \addplot ....
    ...
    \end{loglogaxis}
\end{tikzpicture}
```



5.3 Changing line specifications

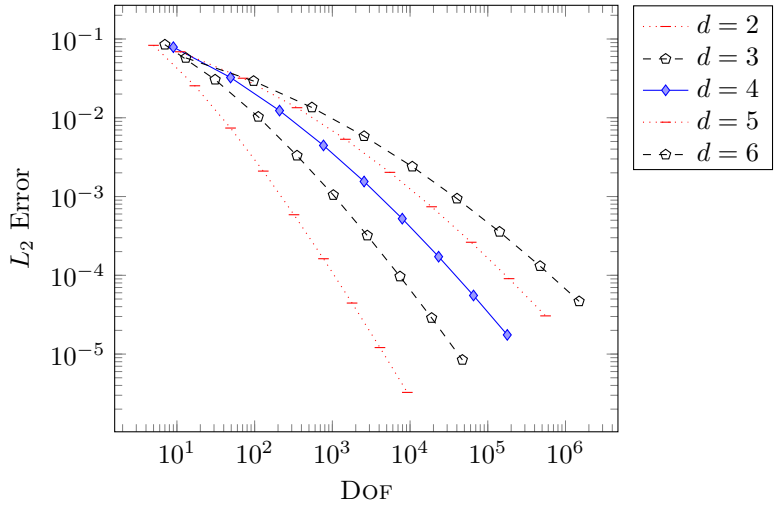
5.3.1 Using another, predefined list

```
\listcopy\blackwhiteplotspeclist\to\autoplotspeclist%
\begin{tikzpicture}
    \begin{loglogaxis}[
        xlabel={\textsc{Dof}},
        ylabel={$L_2$ Error}
    ]
    ...
    \end{loglogaxis}
\end{tikzpicture}
```



5.3.2 Defining new lists

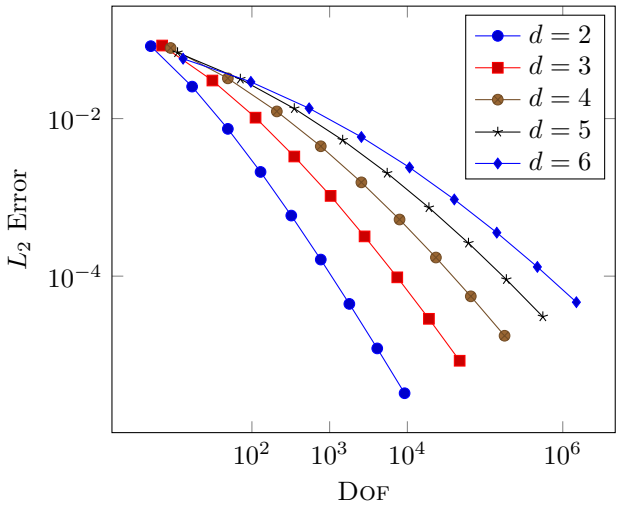
```
% will cycle through these three elements:
\listnew{\autoplotspeclist}{%
    red,dotted,mark=-,mark options={solid}\\%
    black,dashed,mark=pentagon,mark options={solid}\\%
    mark options={fill=blue!40},mark=diamond*,blue\\%
}%
\tikzstyle{every axis legend}+=%
    [at={(1.03,1)},anchor=north west]%
\begin{tikzpicture}
    \begin{loglogaxis}[
        xlabel={\textsc{Dof}},
        ylabel={$L_2$ Error}
    ]
    ...
    \end{loglogaxis}
\end{tikzpicture}
```



5.4 Changing the ticks

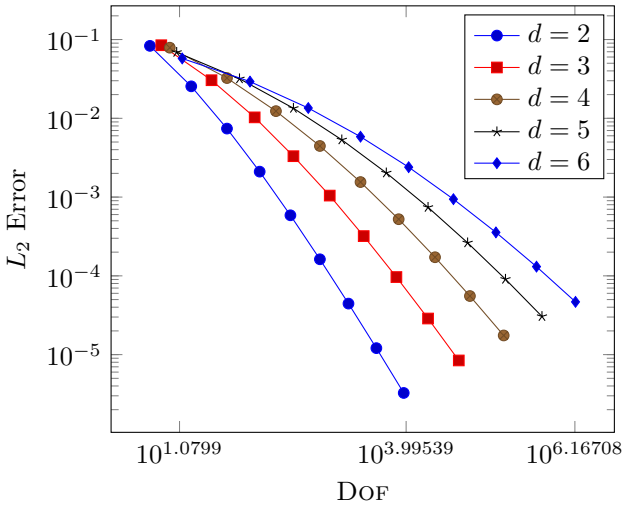
5.4.1 Placing ticks at 10^i

```
\begin{tikzpicture}
    \begin{loglogaxis}[
        xtickten={0,2,3,4,6,...,10},% place tickmarks at
        10^0, 10^2,...
        ytickten={-6,-4,...,2},
        xlabel={\textsc{Dof}},
        ylabel={$L_2$ Error}
    ]
    ....
    \end{loglogaxis}
\end{tikzpicture}
```



5.4.2 Placing ticks anywhere

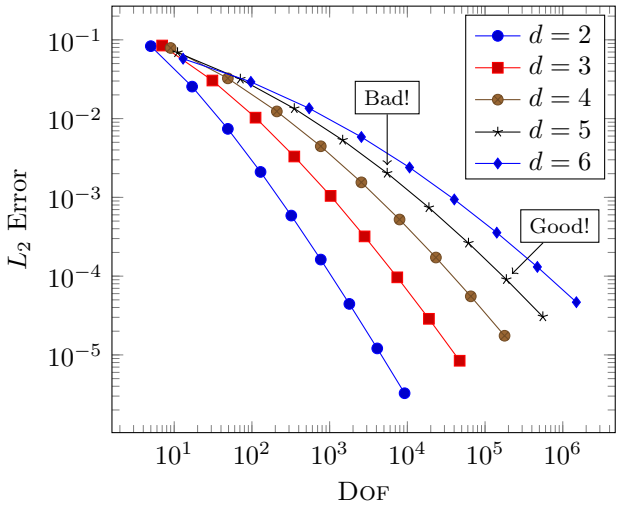
```
\begin{tikzpicture}
    \begin{loglogaxis}[
        xtick={2.5,9.2,14.2},
        xlabel={\textsc{Dof}},
        ylabel={$L_2$ Error}
    ]
    ....
    \end{loglogaxis}
\end{tikzpicture}
```



5.5 Annotating plots

5.5.1 Example: Placing Data Cursors

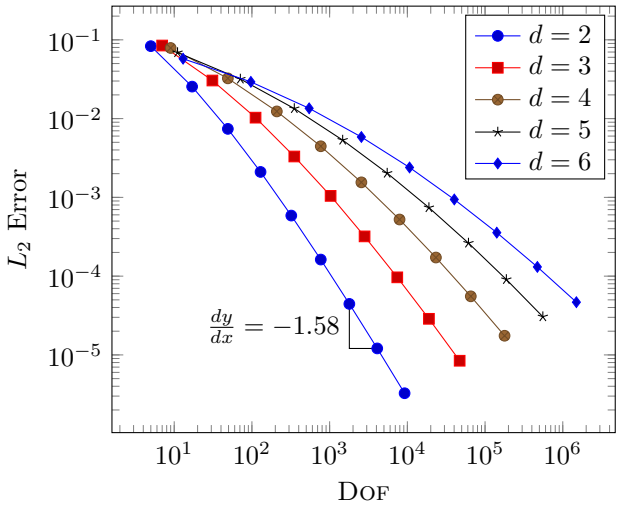
```
\tikzstyle{annotation}=[fill=white,draw=black,font=\
footnotesize]
\tikzstyle{annotedge}=[->,shorten >=3pt]
\begin{tikzpicture}
    \begin{loglogaxis}[
        xlabel={\textsc{Dof}},
        ylabel={$L_2$ Error}
    ]

    ....

    \addplot plot coordinates {
        (13,    5.755e-02)
        (97,    2.925e-02)
        (545,   1.351e-02)
        (2561,  5.842e-03)
        (10625, 2.397e-03)
        (40193, 9.414e-04)
        (141569,    3.564e-04)
        (471041,    1.308e-04)
        (1496065,   4.670e-05)
    };
    ...

    % remember this coordinate under the name 'bad'
    \axispath\node[coordinate] (bad) at
        (8.61305,-6.20135) % = (log(10625), log(2.397e
        -03))
        {};
    \axispath\node[coordinate] (good) at
        (12.14368,-9.30872)
        {};

    % place a node above 'bad' and draw an edge to 'bad':
    \axispath\node[annotation,above of=bad]
        {Bad!}
        edge[annotedge] (bad);
    \axispath\node[annotation,above right of=good]
        {Good!}
        edge[annotedge] (good);
    \end{loglogaxis}
\end{tikzpicture}
```



One remark: the `\axispath` prefix is necessary because neither axis nor plots will be drawn until the x- and y-limits have been determined. All drawing commands are postponed until `\end{axis}` (unless you explicitly provide the options `xmin`, `xmax`, `ymin` and `ymax`).

5.5.2 Example: Slopes of line segments

```
\begin{tikzpicture}
    \begin{loglogaxis}[
        xlabel=\textsc{Dof},
        ylabel=$L_2$ Error
    ]
    \axispath\draw
            (7.49165,-10.02171)
        |-  (8.31801,-11.32467)
        node[near start,left] {$\frac{dy}{dx} = -1.58$};

    \addplot ....
    ...
    \end{loglogaxis}
\end{tikzpicture}
```



References

[1] T. Tantau. Ti*k*Z and PGF manual. `http://tug.ctan.org/cgi-bin/ctanPackageInformation.py?id=pgf`. v. $\geq$ 1.18.