Manual for Package Pgfplots

Version 0.92.4

Christian Feuersänger*
Institut für Numerische Simulation
Universität Bonn, Germany

February 17, 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

*`http://wissrech.ins.uni-bonn.de/people/feuersaenger`

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.

2 Upgrade remarks

This release provides a lot of improvements which can be found in all detail in "`changelog.txt`" for interested readers. However, some attention is required for upgrades:

1. I have re-implemented legends as Ti*k*Z-matrizes and the default legend styles have been changed accordingly.

   Please *replace*

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

   with

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

2. Update each `\axispath...` command in your files. You will need to use `axis cs` as coordinate system to apply any axis transformations. See sections 5.5 and 5.6 for details.

3. The manual file name has been renamed.

3 Installation

3.1 Licensing

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

A copy of the GNU General Public License can be found in the package file

`doc/latex/pgfplots/gpl-3.0.txt`

You may also visit `http://www.gnu.org/licenses`.

3.2 Prerequisites

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

3.3 Files in the Pgfplots-Bundle

The Pgfplots package consists of the files

```
latex/pgfplots/pgfplots.sty
generic/pgfmathlog/pgfmathlog.sty
generic/liststructure/liststructure.sty
generic/sciformat/sciformat.sty
generic/pgfplotshelpers/pgfplotshelpers.sty
doc/latex/pgfplots/pgfplots.bib
doc/latex/pgfplots/pgfplots.*tex
doc/latex/pgfplots/pgfplots.pdf
doc/latex/pgfplots/todo.txt
doc/latex/pgfplots/changelog.txt
test/liststructuretest/liststructuretest.tex
test/pgfplotstest/pgfplotstest.tex
test/pgfmathlogtest/pgfmathlogtest.tex
ext_scripts/pgf2pdf.sh
ext_scripts/matlab2pgfplots.m
ext_scripts/matlab2pgfplots.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.

3.4 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.

3.5 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`.

3.6 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`.

3.7 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.

3.8 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.

4 Drawing axes and plots

4.1 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)
        (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.

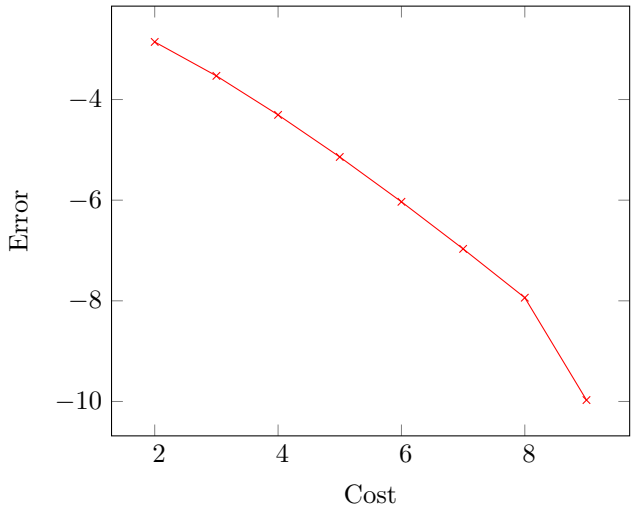


Figure 1: An example for a normal plot. The coordinates are specified using the Ti*k*Z-syntax "`plot coordinates`", optional labels can be provided with the "`xlabel`" and "`ylabel`" arguments.

4.2 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)
        (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)
    };
```

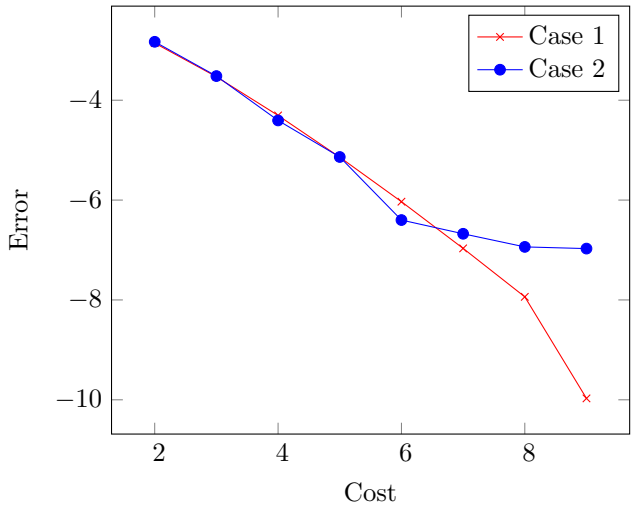


Figure 2: Two plots in the same axis. A legend can be generated using the `\legend`-command.

```
    \legend{Case 1\\Case 2\\}
    \end{axis}
\end{tikzpicture}
```

4.3 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)
    (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}[
```

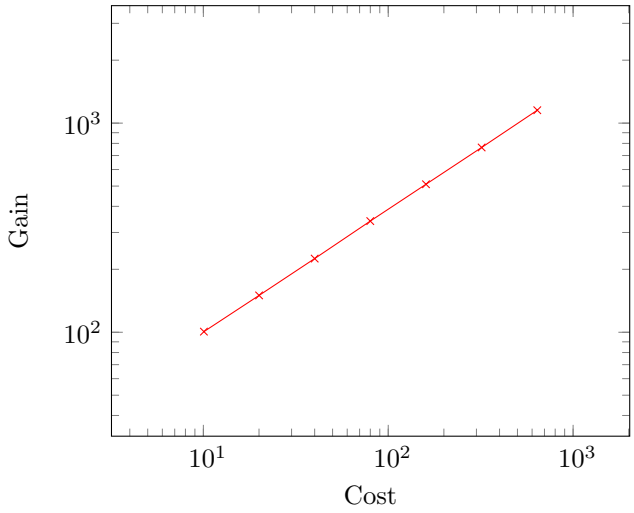


Figure 3: A double–logarithmic plot.

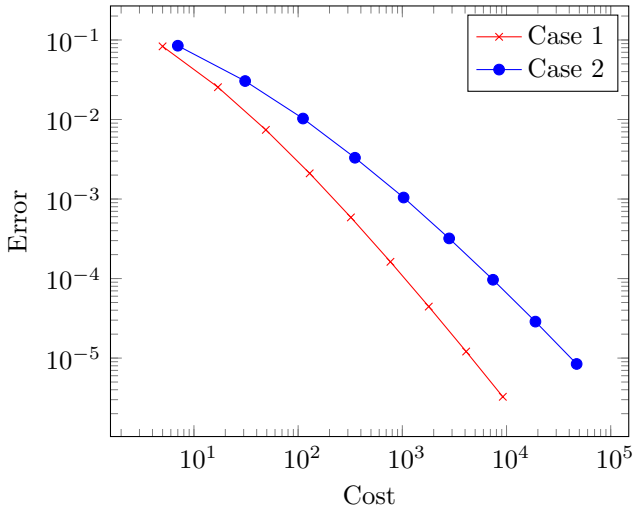


Figure 4: A double–logarithmic plot using scientific notation.

```
        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)
        (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}
    \begin{semilogyaxis}[xlabel=Index,ylabel=Value]
    \addplot[color=blue,mark=*] plot coordinates {
```

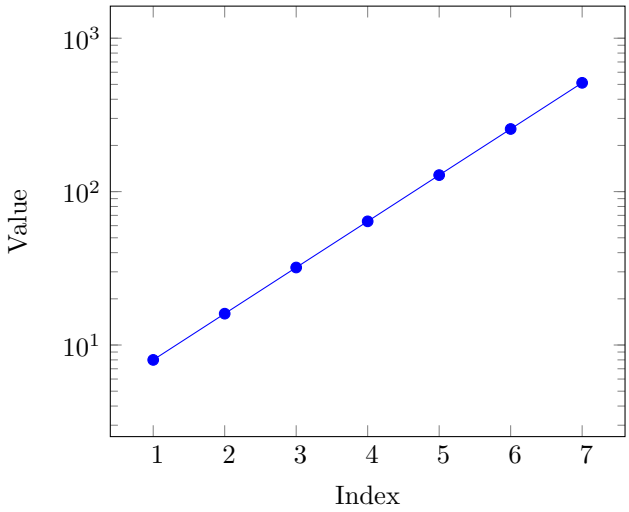


Figure 5: A semi–logarithmic plot.

```
        (1,8)
        (2,16)
        (3,32)
        (4,64)
        (5,128)
        (6,256)
        (7,512)
    };
    \end{semilogyaxis}
\end{tikzpicture}
```

see figure 5.

4.4 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)
        (769,   1.623e-04)
        (1793,  4.442e-05)
```

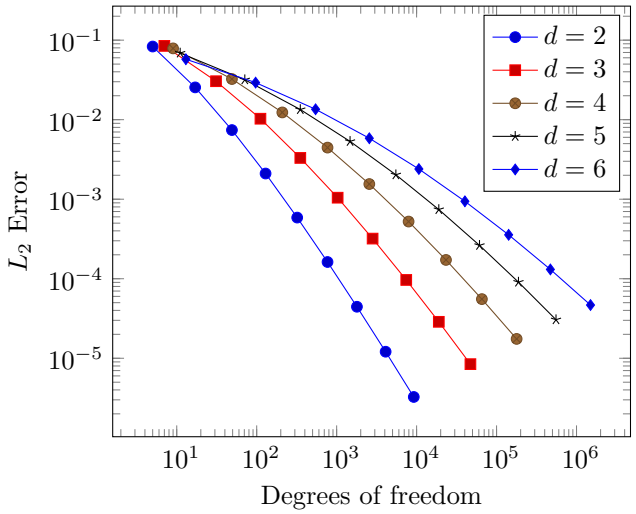
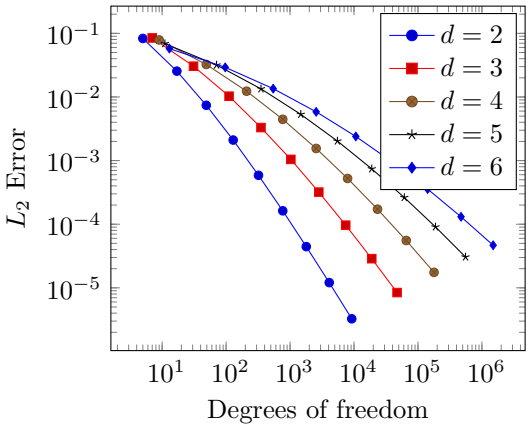


Figure 6: Predefined line/marker combinations.

```
    (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)
```




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

Figure 7: An example of a scaled plot.

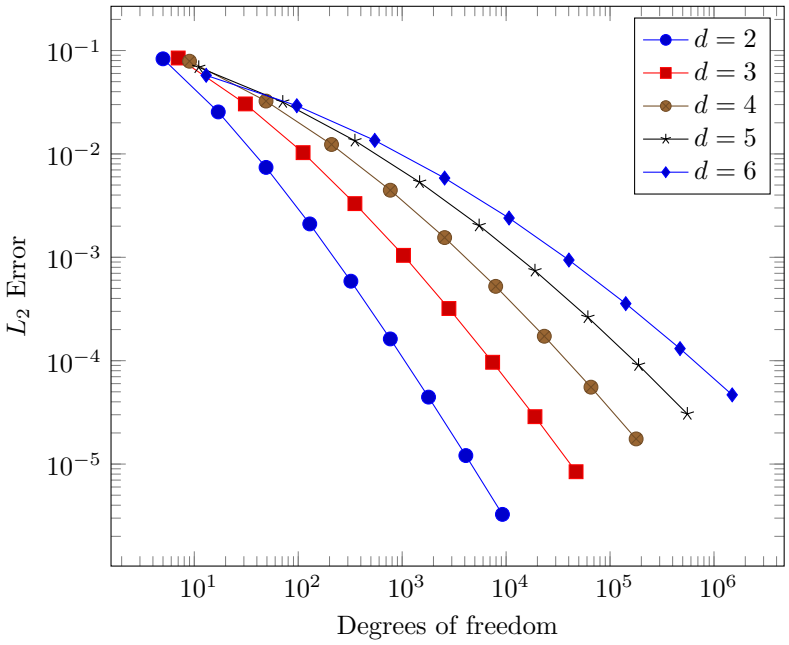
```
        (71,     3.177e-02)
        (351,    1.341e-02)
        (1471,   5.334e-03)
        (5503,   2.027e-03)
        (18943,  7.415e-04)
        (61183,  2.628e-04)
        (187903,     9.063e-05)
        (553983,     3.053e-05)
    };

    \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.

4.5 Scaling plots

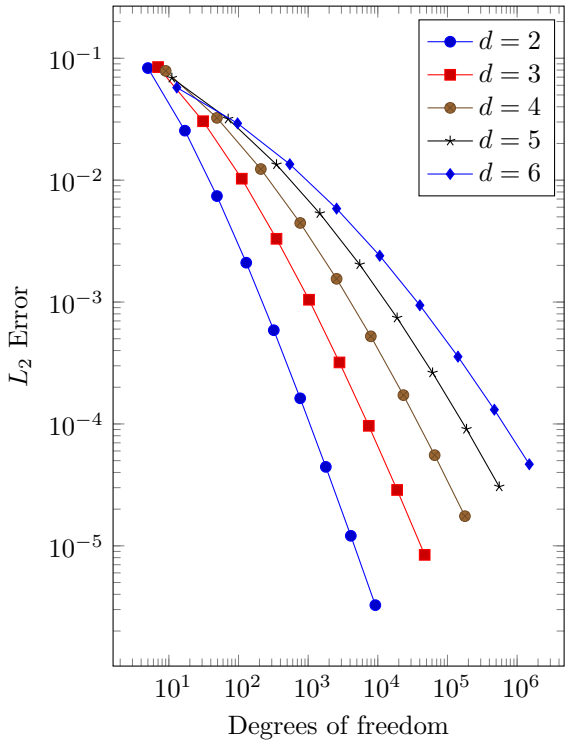
You can use any of the Ti*k*Z options to modify the appearance. For example the effect of the "`scale`" transformation is shown in figures 7 and 8.

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.




```
\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.

More examples can be found in section 6.

5 Reference

5.1 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{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}
```

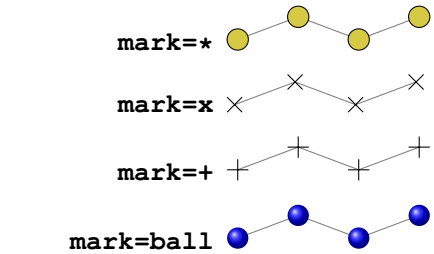
with properly set variables ‘xmode’ and ‘ymode’ (see below).

5.2 Available markers

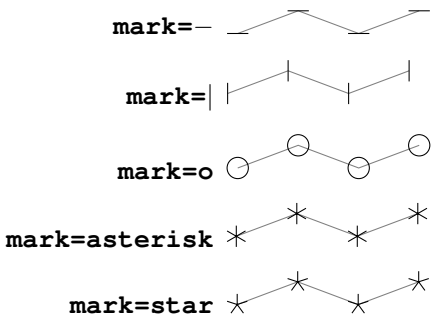
The following options of Ti*k*Z may be useful for plots.

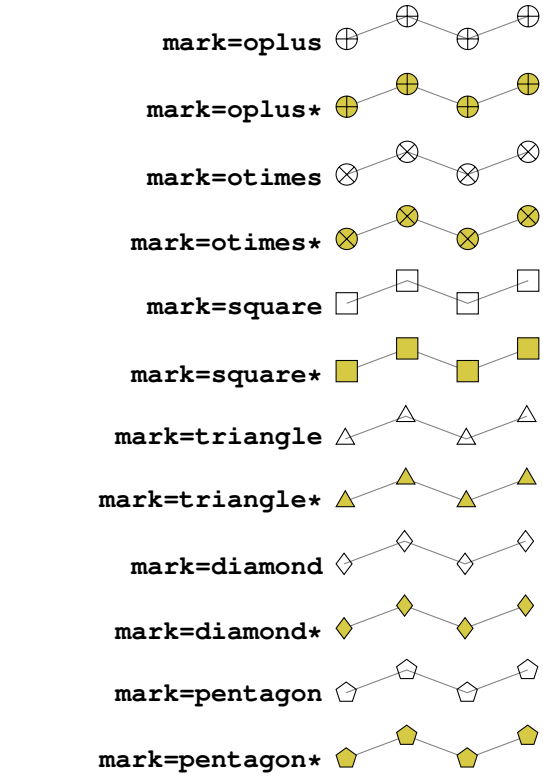
5.2.1 Markers

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
    }
]
```

5.2.2 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.

5.3 \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:

- You can modify `OPTIONS` with

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

  or, even better, with

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

- The `OPTIONS` are remembered for the legend.
- See subsection 5.2 for a list of available markers and line styles.
- 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{tikzpicture}
  \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}
  \end{tikzpicture}
  ```

  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.

- For normal plots, Pgfplots understands arbirtrarily large or small numbers like 0.00000001234 or $1.234 \cdot 10^{24}$. Pgfplots has its own number parsing tools which are complete text based and do not rely on TEX's limited numerical number representation. Before the plots are actually drawn, any input coordinate is transformed into TEX-precision using transformations

  $$T_x(x) = 10^{s_x} \cdot x \text{ and } T_y(y) = 10^{s_y} \cdot y$$

  with properly chosen integers $s_x, s_y \in \mathbb{Z}$. This ensures invariance of axis ranges. See section 5.14.22 for more details.

- As a consequence of the coordinate parsing routines, you can't use the mathematical expression parsing method of PGF as coordinates (that means: you will need to provide coordinates without suffixes like "cm" or "pt" and you can't invoke mathematical functions).

- 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.

5.4 `\addplot+[OPTIONS] ...`

Does the same like `\addplot[OPTIONS] ...` except that `OPTIONS` is *appended* to the arguments which would have been taken for `\addplot ...` (the element of the default list).

Example:

```
\addplot+[only marks] plot coordinates {...};
```

will use the same line style, markers, colors as

```
\addplot plot coordinates {...};
```

but it will only draw markers, no lines.

5.5 `\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 (axis cs:12.14368,-9.30872) {};
```

or

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

A useful example is presented in section 6.5 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.

The "`axis cs`" coordinate system allows you to access axis coordinates. You can use the same numbers as in `\addplot` all required transformations like logarithms or data scaling transformations will be applied. All `\axispath` commands should use these coordinate systems. See section 5.6 for more information and section 6.5 for examples.

5.6 Accessing axis coordinates with `axis cs`

Pgfplots provides a new coordinate system for use inside of an axis, the "axis coordinate system", `axis cs`.

Its can be used in conjunction with `\axispath` to draw any Ti*k*Z-graphics at axis coordinates. It is used like

```
\axispath\draw
    (axis cs:18943,2.873391e-05)
|-  (axis cs:47103,8.437499e-06);
```

see section 6.5 for example graphics.

Attention: Whenever you draw additional graphics, consider using `axis cs`! It applies any logarithms, data scaling transformations or whatever Pgfplots usually does!

5.7 `\addlegendentry{name}`

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

```
\begin{tikzpicture}
\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)
```

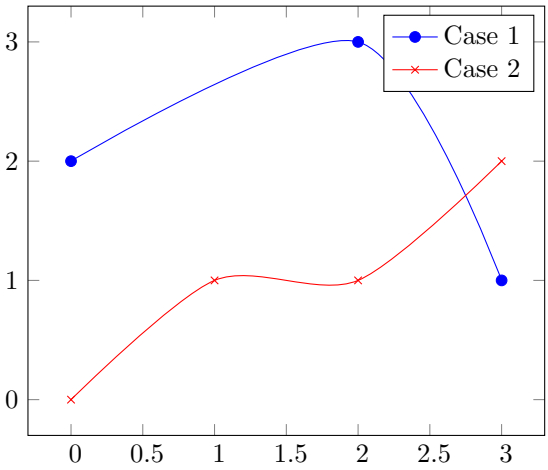


Figure 10: An example for the \addlegendentry command

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

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.

5.8 \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. **Attention:** the final \\ is necessary! The \legend command takes the following arguments:

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

where each legend element needs to be terminated by \\. The short marker/line combination shown in legends is acquired from the argument to \addplot[...].

5.8.1 Legend appearance

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

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

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

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

means the upper right corner. The 'anchor' option determines which point *of the legend* will be placed at $(0,0)$ or $(1,1)$ (see below for more examples).

The legend is a Ti*k*Z-matrix, so you can use any Ti*k*Z option which affects nodes and matrizes (see [1, section 13 and 14]). The matrix is created by something like

```
\matrix[style=every axis legend] {
    draw plot specification 1 & \node{legend 1}\\
    draw plot specification 2 & \node{legend 2}\\
    ...
};
```

You can configure the number of horizontal legend entries with the axis-option "legend columns=NUMBER". For example,

```
\begin{tikzpicture}
\begin{axis}[legend columns=4]
...
\legend{legend 1\\legend 2\\legend 3\\}
\end{axis}
\end{tikzpicture}
```

would use (up to) 4 adjacent legend entries.

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}=[%
        cells={anchor=center},
        inner xsep=3pt,inner ysep=2pt,nodes={inner sep=2
        pt,text depth=0.15em},
        anchor=north east,%
        shape=rectangle,%
        fill=white,%
        draw=black,
        at={(0.98,0.98)}
        ]
```

Attention: you should *not reset* the default style to stay compatible with future versions. If possible, use

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

5.9 \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.

5.10 \logtologten{ARG}

Expands to the result of `ARG`/ log(10). You can also use

```
\logtologtentomacro{9.5}{\result}
```

and `\result` will be filled with the result.

5.11 \logten

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

5.12 \prettyprintnumber{NUM}

Allows to generate pretty–printed output for the number `NUM`. Examples:

```
\prettyprintnumber{1.0}

\prettyprintnumber{151.015341}

\prettyprintnumber{90.99999}
```

results in
1,
151.02,
91.
It is used to display axis ticks. The default rounding precision is defined by

```
\def\prettyprintnumberprecision{2}
```

and can be reconfigured if needed (the highest precision is 5 due to TEX's restricted numerical capabilities).

5.13 \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.

5.14 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.

5.14.1 [xy]min=COORD, [xy]max=COORD

The options xmin, xmax and ymin, ymax allow to define 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.

5.14.2 [xy]label=TEXT

The options xlabel and ylabel change 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]
```

You should use

```
\tikzstyle{every axis label}+=[...]
\tikzstyle{every axis x label}+=[...]
\tikzstyle{every axis y label}+=[...]
```

to modify options to ensure compatibility with future versions.

5.14.3 `title=TEXT`

Adds a caption to the plot. This will place a Ti*k*Z-Node with

```
\node[style=every axis title] {TEXT};
```

to the current axis. An example is shown in figure 11. The title is placed in the middle of the axis (the placing does not incorporate any axis descriptions). You can reconfigure the appearance and/or placing of the title for example with

```
\tikzstyle{every axis title}+=[at={(0.75,1)}]
```

This will place the title at 75% of the x-axis. The coordinate $(0, 0)$ is the lower left corner and $(1, 1)$ the upper right one.

Please note that using the LaTeX floating figures together with `\caption{TEXT}` may be more appropriate in many situation. However, '`title`' allows access to the axis' size without any axis descriptions which is useful if more than one plot needs a caption.

5.14.4 `[xy]mode=normal|log`

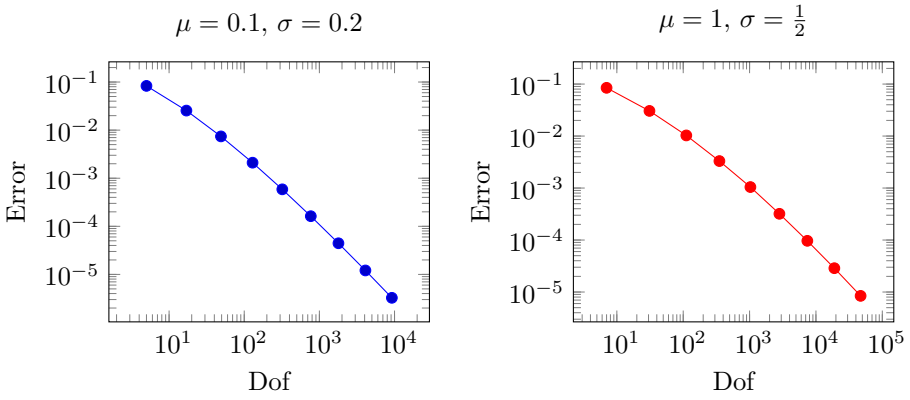
Allows to choose between normal plots or logplots for each x, y-combination. Default is `xmode=normal`, `ymode=normal`.

5.14.5 `[xy]tick={LIST}`

The options `xtick` and `ytick` 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.
- Use '`xtick=`' to use the default tick placement rules.
- Use '`xtick=\empty`' to disable any ticks.

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




```
\begin{tikzpicture}
\begin{loglogaxis}[
    width=0.48\textwidth,
    xlabel=Dof,ylabel=Error,
    title={$\mu=0.1$, $\sigma=0.2$}]

    \addplot plot coordinates {
        (5,     8.312e-02)
        (17,    2.547e-02)
        ...
        (4097,  1.207e-05)
        (9217,  3.261e-06)
    };
\end{loglogaxis}
\end{tikzpicture}%
\hfill
\begin{tikzpicture}
\begin{loglogaxis}[
    width=0.48\textwidth,
    xlabel=Dof,ylabel=Error,
    title={$\mu=1$, $\sigma=\frac{1}{2}$}]

    \addplot[color=red,mark=*] plot coordinates {
        (7,     8.472e-02)
        (31,    3.044e-02)
        ...
        (18943, 2.873e-05)
        (47103, 8.437e-06)
    };
\end{loglogaxis}
\end{tikzpicture}
```

Figure 11: An example for the 'title' option. Some data points have not been listed, the '...' is not part of the plot.

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 tick *positions* in normal plots is to place a tick at each coordinate $i \cdot h$. The step size h depends on the axis scaling and the axis limits. It is chosen from a list a "feasable" step sizes such that neither too much nor too few ticks will be generated. The default for logplots it to place ticks at each 10^i in the axis' range. The default tick positions can be reconfigured with

- `\renewcommand{\axisdefaulttickwidth}{35}` where the integer argument denotes the maximum space between adjacent ticks in full points. The suffix "pt" has to be omitted and fractional numbers are not supported.
- `\renewcommand{\axisdefaulttryminticks}{4}` configures a loose lower bound on the number of ticks. It should be considered as a suggestion, not a tight limit.

The tick *appearance* can be (re-)configured with

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

or

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

Please prefer the '+=' versions to ensure compatibility with future versions.

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

5.14.6 [xy]tickten={LIST}

The options `xtickten` and `ytickten` 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`'.

5.14.7 [xy]ticklabels={LIST}

Assigns a *list* of tick *labels* to each tick position. Tick *positions* are assigned using the `xtick` and `ytick`-options.

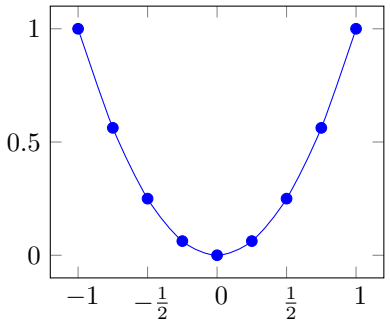
This is one of two options to assign tick labels directly. The other option is '`xticklabel={COMMAND}`' (or `yticklabel={COMMAND}`). Option '`ticklabel`' offers higher flexibility while '`ticklabels`' is easier to use.

The argument `LIST` has the same format as for ticks, that means

```
xticklabels={$\frac{1}{2}$,$e$}
```

Denotes the two–element–list $\{\frac{1}{2}, e\}$. The list indices match the indices of the tick positions. If you need commas inside of list elements, use

```
xticklabels={{0,5}, $e$}.
```




```
xtick={-1.5,-1,...,1.5},
xticklabels={%
    $-1\frac 12$,
    $-1$,
    $-\frac 12$,
    $0$,
    $\frac 12$,
    $1$}
```

Figure 12: An example for the `xticklabels` option.

Example:

```
\begin{tikzpicture}
\begin{axis}[
    xtick={-1.5,-1,...,1.5},
    xticklabels={%
        $-1\frac 12$,
        $-1$,
        $-\frac 12$,
        $0$,
        $\frac 12$,
        $1$}
]
\addplot[smooth,blue,mark=*] plot coordinates {
    (-1,    1)
    (-0.75, 0.5625)
    (-0.5,  0.25)
    (-0.25, 0.0625)
    (0,     0)
    (0.25,  0.0625)
    (0.5,   0.25)
    (0.75,  0.5625)
    (1,     1)
};
\end{axis}
\end{tikzpicture}
```

yields figure 12.

5.14.8 `[xy]ticklabel={COMMAND}`

Use `xticklabel` or `yticklabel` to change the LaTeX-command which creates the tick *labels* assigned to each tick position (see options `xtick` and `ytick`).

This is one of two options to assign tick labels directly. The other option is '`xticklabels={LIST}`' (or `yticklabels={LIST}`). Option '`ticklabel`' offers higher flexibility while '`ticklabels`' is easier to use.

The argument '`COMMAND`' can be any LaTeX-text. The following commands are valid inside of `COMMAND`:

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

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

The default argument is

- `\axisdefaultticklabel` for normal plots and
- `\axisdefaultticklabellog` for logplots, see below.

(the same holds for `yticklabel`). The defaults are set to

```
\newcommand{\axisdefaultticklabel}[0]{%
    $\prettyprintnumber{\tick}$%
}

\newcommand{\axisdefaultticklabellog}[0]{{%
    \logtologtentomacro{\tick}{\roundedtick}%
    $10^{\prettyprintnumber{\roundedtick}}$%
}}
```

You can change the appearance of tick labels with

```
\tikzstyle{every tick label}+=[font=\tiny]
```

and/or

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

and

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

5.14.9 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.

5.14.10 [xy]minorticks=true|false

5.14.11 [xy]majorticks=true|false

5.14.12 ticks=minor|major|both|none

Enables/disables the small tick lines either for single axis or for all of them. Major ticks are those placed at the tick positions and minor ticks are only used in log plots. Please note that minor ticks are automatically disabled if `xtick` is not a uniform range[1]. Default is to use minor and major ticks.

You can configure the length of the tick line with '`minor tick length=DIMEN`' and '`major tick length=DIMEN`' (where DIMEN is a TEX-length like 1cm) and its appearance using the following styles:

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

```
\tikzstyle{every tick}+=[color=black] % applies to major
and minor ticks,
\tikzstyle{every minor tick}+=[thin]  % applies only to
minor ticks,
\tikzstyle{every major tick}+=[thick] % applies only to
major ticks.
```

There is also the style "`every tick`" which applies to both, major and minor ticks.

5.14.13 major tick length=DIMEN

5.14.14 minor tick length=DIMEN

Allows to configure the length of the small tick lines. Minor ticks are only displayed for logarithmic plots. See option "`ticks`" for appearance styles for ticks.

5.14.15 [xy]minorgrids=true|false

5.14.16 [xy]majorgrids=true|false

5.14.17 grid=minor|major|both|none

Enables/disables different grid lines. Major grid lines are placed at the normal tick positions (see `xmajorticks`) while minor grid lines are placed at minor ticks (see `xminorticks`).

Grid lines will be drawn before tick lines are processed, so ticks will be drawn on top of grid lines. You can configure the appearance of grid lines with the styles

```
\tikzstyle{every axis grid}=[style=help lines] % applies
to major and minor grids,
\tikzstyle{every minor grid}+=[color=blue]     % applies
only to minor grid lines,
\tikzstyle{every major grid}+=[thick]          % applies
only to major grid lines.
```

An example for grid lines can be found in section 6.8.

5.14.18 enlargelimits=[true|false|auto|VAL]

Enlarges the axis size somewhat if enabled.

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.

- The value `true` enlarges all axes.
- The value `false` uses tight axis limits as specified by the user (or read from input coordinates).
- The value `auto` will enlarge limits only for axis for which axis limits have been determined automatically.

- All other values like 'enlargelimits=0.1' will enlarge all axis limits relatively (in this example, 10% of the axis limits will be added at all sides).
- If no relative threshold is provided, axis enlargement is done relatively to $x_{\max} - x_{\min}$ for normal plots and absolutely for log–plots up to now.

A small value of enlargelimits may avoid problems with large markers near the boundary.

5.14.19 legend columns=NUMBER

Allows to configure the maximum number of adjacent legend entries. The default is legend columns=1, so legends are placed vertically below each other. Examples can be found in section 6.1.4.

5.14.20 legend plot pos=left|right|none

Configures where the small line specifications will be drawn: left of the description, right of the description or not at all. See an example in section 6.1.5.

5.14.21 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).

5.14.22 disabledatascaling

Disables internal re-scaling of input data. Normally, every input data like plot coordinates, tick positions or whatever, are parsed without using TeX's limited number precision. Then, a transformation like

$$T(x) = 10^{q-m} \cdot x$$

is applied to every input coordinate/position where m is "the order of x" base 10. Example: $x = 1234 = 1.234 \cdot 10^3$ has order $m = 4$ while $x = 0.001234 = 1.234 \cdot 10^{-3}$ has order $m = -2$. The parameter q is the order of the axis' width/height.

The **effect** is that your plot coordinates can be of *arbitrary precision* like 0.0000001 and 0.0000004. For these two coordinates, Pgfplots will use 100pt and 400pt internally. The transformation is quit fast since it relies only on period shifts.

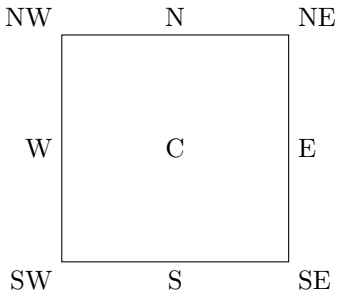
The option "`disabledatascaling`" disables this data transformation. That means you are restricted to coordinates which TEX can handle[2].

So far, the data scale transformation applies only to normal axis (logarithmic scales do not need it).

5.14.23 `anchor=NAME`

This option shifts the axis horizontally and vertically such that the axis anchor (a point on the axis) is placed at coordinate $(0, 0)$[3].

- Anchors are useful in conjunction with horizontal or vertical alignment of plots, see the examples in section 6.6 and section 6.7.
- Anchors are placed on the axis rectangle; any axis labels, titles or legends are completely unaffected.
- Valid choices for `NAME` are `north`, `north west`, `west`, `south west`, `south`, `south east`, `east`, `north east`, `center`. They denote the places



- The default value is `anchor=south west`.

5.14.24 `width=DIMEN`, `height=DIMEN`

The axis is always scaled such that its dimension is

```
(\axisdefaultwidth,\axisdefaultheight).
```

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:

[2]Please note that the axis' scaling requires to compute $1/(x_{\max} - x_{\min})$. The option `disabledatascaling` may lead to overflow or underflow in this context, so use it with care! Normally, the data scale transformation avoids this problem.

[3]I haven't understood how to implement an axis as a path command. In the future, the last path coordinate may be used in this place.

- 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, but their size is used to determine the axis scaling.
- 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.
- You can also force a fixed width/height of the axis (without looking at labels) with

```
\begin{tikzpicture}
\begin{axis}[width=5cm,scale only axis]
    ...
\end{axis}
\end{tikzpicture}
```

- 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[4]. This may be fixed in future versions.

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

5.14.25 scale only axis=[true|false]

This boolean option allows to apply `width` and `height` only to the axis rectangle. If '`scale only axis`' is enabled, label, tick and legend dimensions won't influence the size of the axis rectangle.

If `scale only axis=false` (the default), Pgfplots will try to produce the desired width *including* labels, titles and ticks.

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

5.14.26 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.

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).

5.14.27 hide axis=[true|false]

Allows to hide the axis. No outer rectangle, no tick marks and no labels will be drawn. Only titles and legends will be processed as usual.

Axis scaling and clipping will be done as if you did not use `hide axis`.

5.14.28 [xy]filter=CMD

The option `xfilter` and `yfilter` 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.

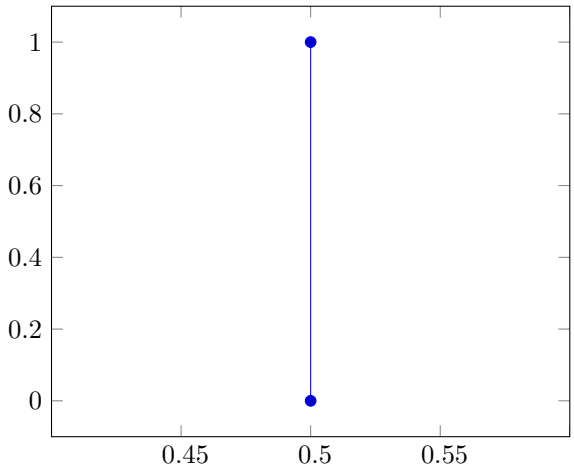
Be warned: '`xfilter`' and '`yfilter`' are advanced options and may require advanced knowledge about TEX.

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`.
- The arguments to coordinate filters are not transformed. You may need to call coordinate parsing routines.
- 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.

5.15 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.

5.16 execute at end plot=COMMANDS

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

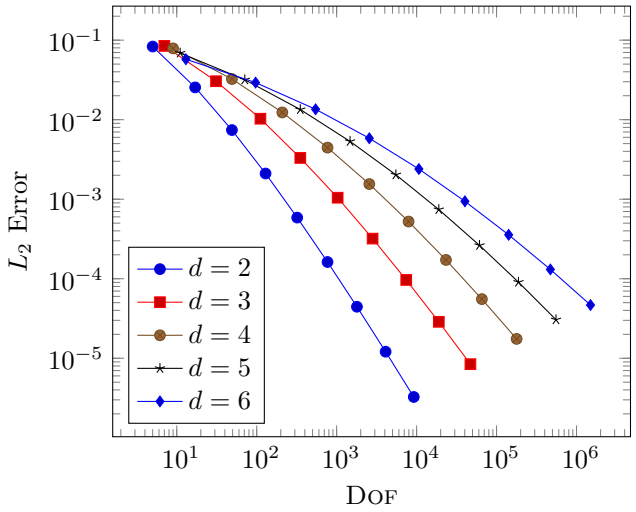
6 More examples

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

6.1 Legend position and appearance

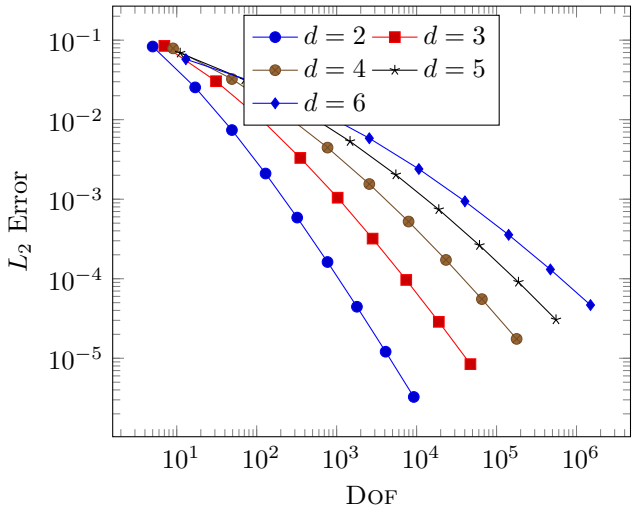
6.1.1 Legend in the lower left corner

```
\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}
```



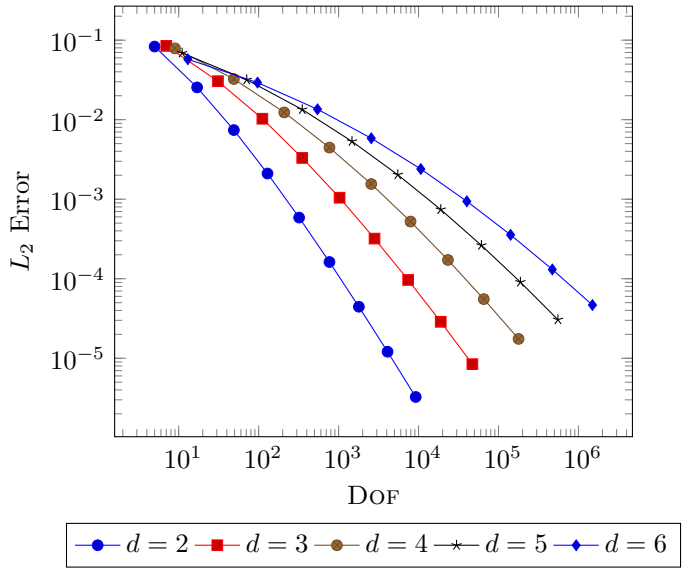
6.1.2 Horizontal Legends

```
\tikzstyle{every axis legend}+=
    [at={(0.5,0.98)},anchor=north]%
\begin{tikzpicture}
    \begin{loglogaxis}[
        legend columns=2,
        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}
```



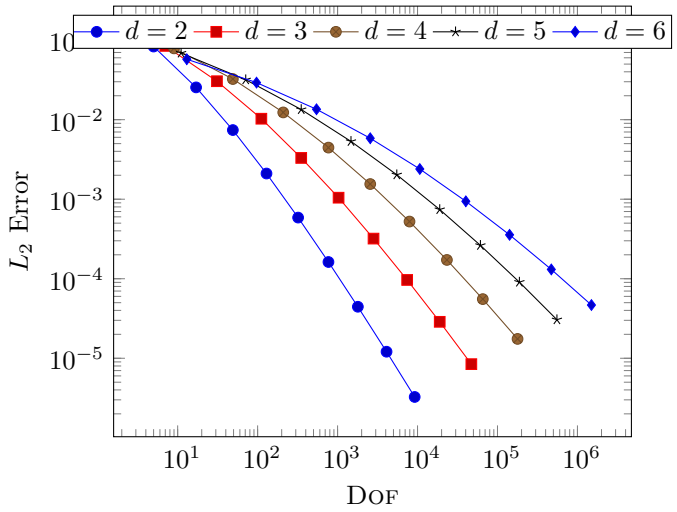
6.1.3 Horizontal Legends (2)

```
\tikzstyle{every axis legend}+=
    [at={(0.5,-0.2)},anchor=north]%
\begin{tikzpicture}
    \begin{loglogaxis}[
        legend columns=-1,
        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}
```



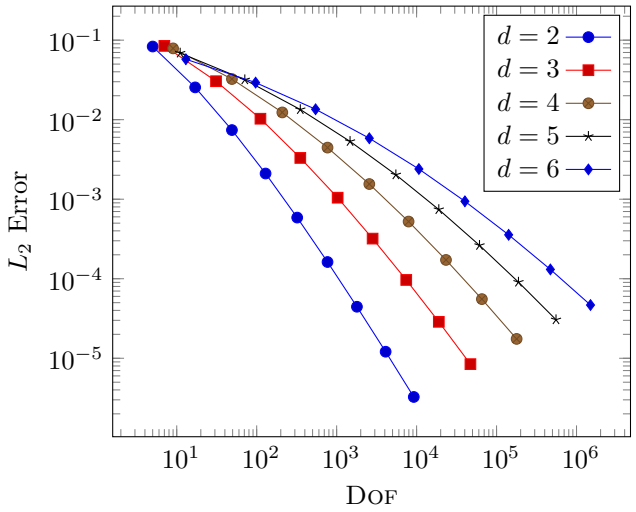
6.1.4 Modifying legend spacing

```
\tikzstyle{every axis legend}+=
    [at={(0.5,1.02)},anchor=south,inner sep=0pt]%
\begin{tikzpicture}
    \begin{loglogaxis}[
        legend columns=-1,
        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}
```



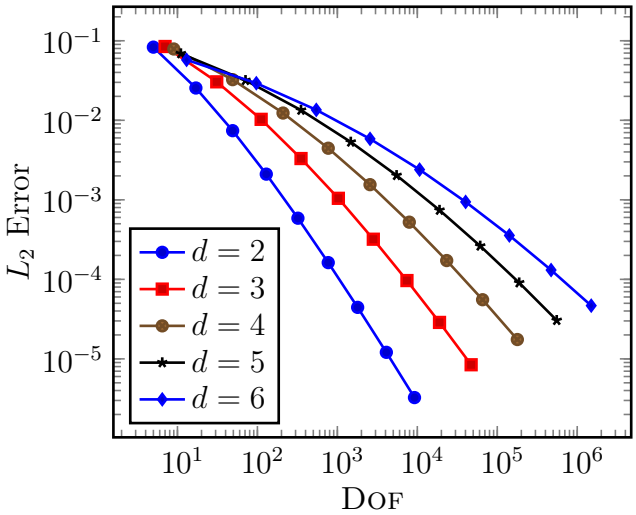
6.1.5 Modifying legend's small plots

```
\begin{tikzpicture}
    \begin{loglogaxis}[
        legend plot pos=right,
        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}
```



6.2 Font size and line width

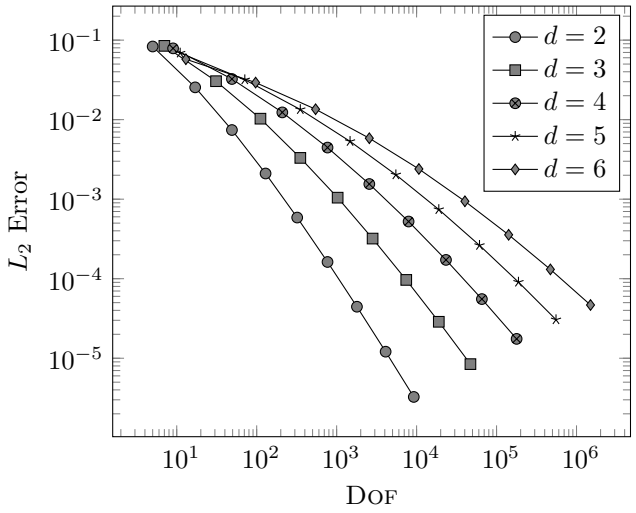
```
\tikzstyle{every axis legend}+=%
    [at={(0.03,0.03)},anchor=south west]%
\tikzstyle{every tick}+=[line width=0.6pt]%
\begin{tikzpicture}[font=\large,line width=1pt]
    \begin{loglogaxis}[
        xlabel={\textsc{Dof}},
        ylabel={$L_2$ Error}
    ]
    \addplot ....
    ...
    \end{loglogaxis}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



6.3 Changing line specifications

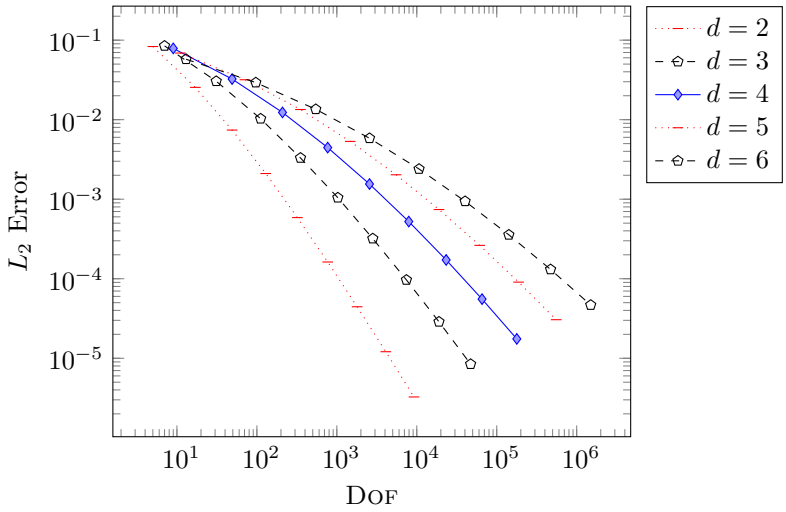
6.3.1 Using another, predefined list

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



6.3.2 Defining new lists

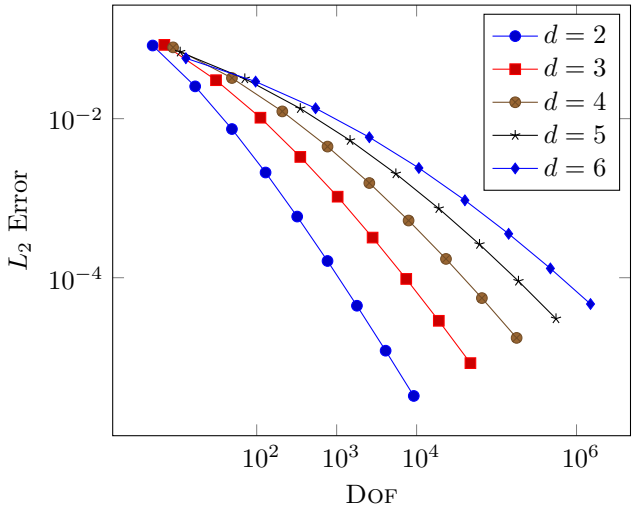
```
% will cycle through these three elements:
\listnew{\autoplotspeclist}{%
    red,dotted,mark=-,mark options={solid}\\%
    black,dashed,mark=pentagon,mark options={solid}\\%
    mark options={fill=blue!40},mark=diamond*,blue\\%
}%
\tikzstyle{every axis legend}+=%
    [at={(1.03,1)},anchor=north west]%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



6.4 Changing the ticks

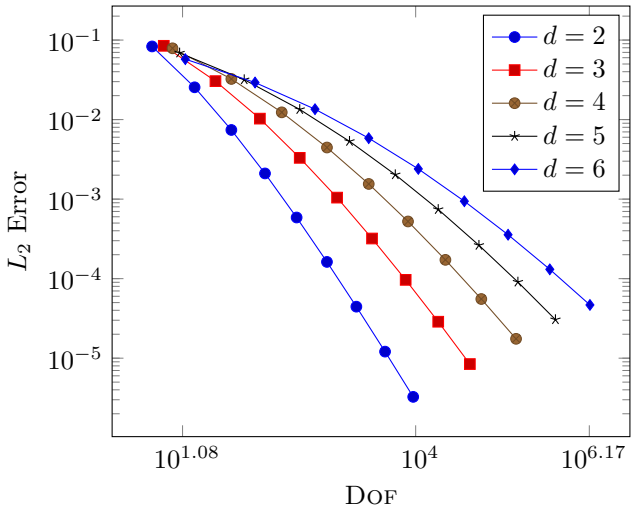
6.4.1 Placing ticks at 10^i

```
\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}
```



6.4.2 Placing ticks anywhere

```
\begin{tikzpicture}
    \begin{loglogaxis}[
        xtick={12,9897,1468864}
        xlabel={\textsc{Dof}},
        ylabel={$L_2$ Error}
    ]
    ....
    \end{loglogaxis}
\end{tikzpicture}
```

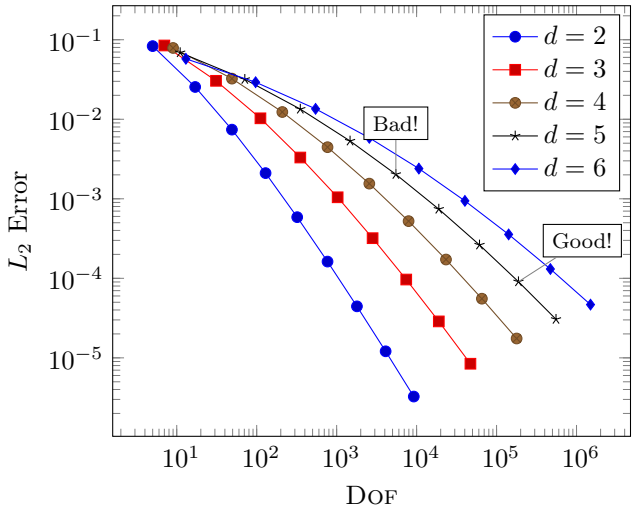


6.5 Annotating plots

6.5.1 Example: Placing Data Cursors

```
\tikzstyle{every pin}=[fill=white,draw=black,font=\
footnotesize]
\begin{tikzpicture}
    \begin{loglogaxis}[
        xlabel={\textsc{Dof}},
        ylabel={$L_2$ Error}
    ]

    ....

    \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)
    };
    ...

    \axispath\node[coordinate,pin=above:Bad!]
        at (axis cs:5503,2.027e-03) {};
    \axispath\node[coordinate,pin=above right:Good!]
        at (axis cs:187903,9.063e-05)   {};
    \end{loglogaxis}
\end{tikzpicture}
```

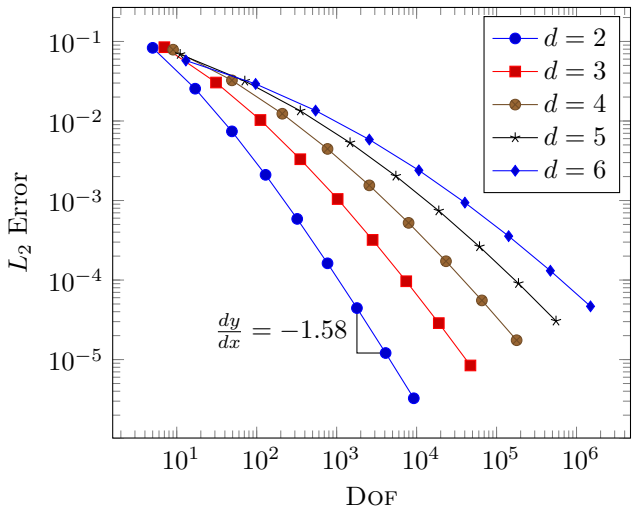


One remark: the `\axispath` prefix is necessary because neither axis nor plots will be drawn until the x- and y-limits have been determined. All drawing commands are postponed until `\end{axis}` (unless you explicitly provide the options `xmin`, `xmax`, `ymin` and `ymax`).

6.5.2 Example: Slopes of line segments

```
\begin{tikzpicture}
    \begin{loglogaxis}[
        xlabel=\textsc{Dof},
        ylabel=$L_2$ Error
    ]
    \axispath\draw
            (axis cs:1793,4.442e-05)
        |-  (axis cs:4097,1.207e-05)
        node[near start,left] {$\frac{dy}{dx} = -1.58$};

    \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)
    };
    ...
    \end{loglogaxis}
\end{tikzpicture}
```



6.6 Vertical alignment with the `baseline` option

The default axis anchor is `south west`, which means that the picture coordinate $(0, 0)$ is the lower left corner of the axis. As a consequence, the Ti*k*Z option "`baseline`" allows vertical alignment of adjacent plots, see figure 13 for an example.

6.7 Horizontal Alignment

The last subsection used the `baseline`-option to allow *vertical* alignment of plots. You can also apply *horizontal* alignment, if you place multiple `axes` into a single `tikzpicture` and use the '`anchor`'-option:

```
\begin{tikzpicture}
    \begin{axis}[width=4cm,scale only axis]
    \addplot[red,only marks,mark options={fill=red,scale
    =0.8},mark=*] plot coordinates {
        (0.190600,  0.860900)
        (0.194400,  0.908000)
        (0.179900,  0.140700)
        (0.212400,  0.294200)
        ....
```

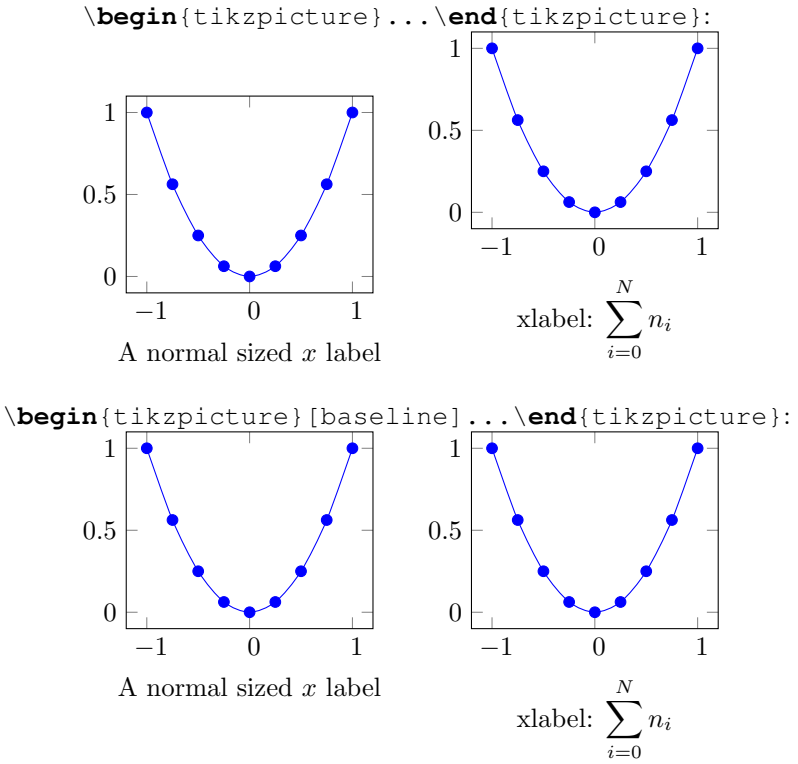


Figure 13: An example of the Ti*k*Z-"`baseline`" option for vertical alignment. Top row: without "`baseline`", bottom row: with "`baseline`".

```
    (0.202300,  0.222300)
    (0.193300,  0.959700)
};
\addplot[black,only marks,mark options={fill=black,
scale=0.8},mark=square*] plot coordinates {
    (0.005300,  0.369500)
    (-0.016100, 0.610000)
    (-0.007400, 0.252100)
    (0.002900,  0.699600)
    (0.010900,  0.232000)
    ....
    (-0.003600, 0.068000)
    (-0.022400, 0.575400)
    (-0.002900, 0.184500)
};
\addplot[blue] plot coordinates {
    (0.093947,  -0.011481)
    (0.101957,  0.494273)
    (0.109967,  1.000027)
};
\end{axis}

\draw[thick,gray,->]
    (0,0) -- (0cm,-0.6cm)
    node[black,midway,left] {0.6cm$\to$};

\begin{scope}[yshift=-0.6cm]
\begin{axis}[%
    anchor=north west,
    width=4cm,scale only axis,
    height=0.8cm,
    ytick=\empty
]

\addplot[red,only marks,mark options={fill=red,scale
=0.8},mark=*] plot coordinates {
    (0.968555,  0)
    (0.030884,  0)
    ...
    (0.468750,  0)
    (0, 0)
};
\addplot[black,only marks,mark options={fill=black,
scale=0.8},mark=square*] plot coordinates {
    (0.367188,  0)
    (0.312500,  0)
    ...
    (0.125000,  0)
    (0.156250,  0)
};
```

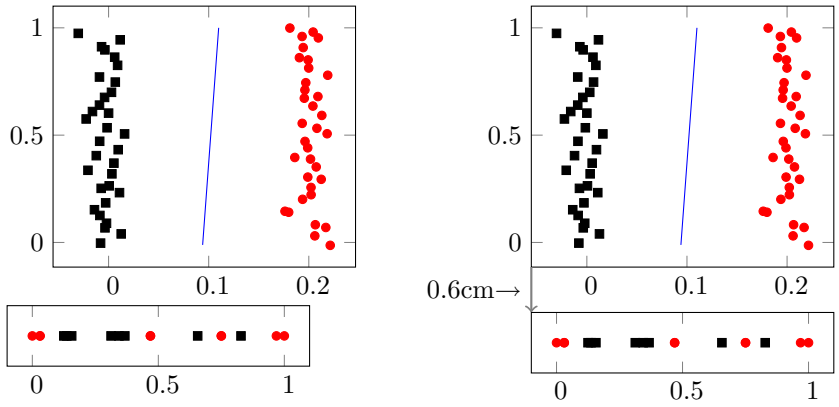


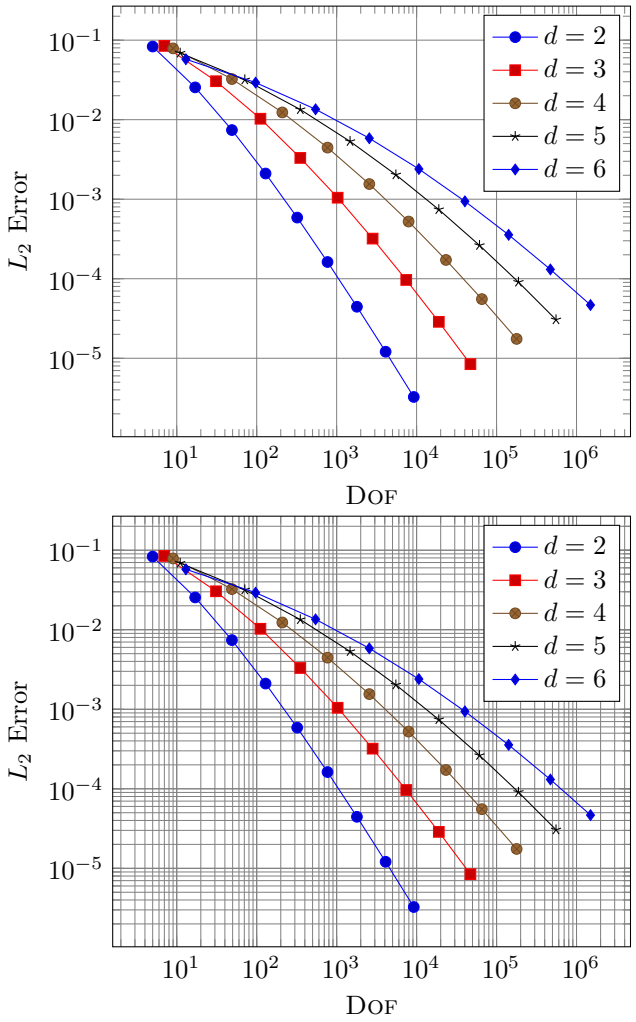
Figure 14: Demonstration of horizontal alignment with the 'anchor' option. Left: two tikzpictures placed above each other; right: one tikzpicture with horizontal alignment using the 'anchor'-option. The source code is in the text.

```
    \end{axis}
    \end{scope}
\end{tikzpicture}
```

The result of this alignment is shown in figure 14.

6.8 Gridlines

```
\begin{tikzpicture}
    \begin{loglogaxis}[
        xlabel={\textsc{Dof}},
        ylabel={$L_2$ Error},
        grid=major
    ]
    ...
    \end{loglogaxis}
\end{tikzpicture}

\begin{tikzpicture}
    \begin{loglogaxis}[
        xlabel={\textsc{Dof}},
        ylabel={$L_2$ Error},
        grid=both
    ]
    ...
    \end{loglogaxis}
\end{tikzpicture}
```



7 Import/Export from other formats

There are two (experimental) scripts which may be used to generate LaTeX-code with plot coordinates or to simplify format conversion. They are not necessary to run Pgfplots.

7.1 pgf2pdf.sh

This is a unix `bash`–script which attempts to simplify the workflow of creating PDF-Dokuments for single plots.

PGF provides a method to externalize graphics, meaning to write specific portions of the LaTeX-file to separate PDF/EPS-files. The script pgf2pdf.sh assumes that portions which need externalization are in files '`.pgf`' and should be converted to PDF/EPS. Two ways are supported:

1. Run LaTeX on the complete document and write only the portion of interest,

2. Run LaTeX *only* on the '`.pgf`'-file. This mode requires a TeX-file which defines all required commands and includes all required packages (a header).

If you are interested in externalized graphics, you should read[1, Section 54]. You may also type

```
pgf2pdf.sh --help.
```

This script is experimental.

7.2 matlab2pgfplots.m

This is a matlab (tm) script which attempts to convert a matlab figure to Pgfplots.

The idea is to

- use a complete matlab figure as input,
- acquire axis labels, axis scaling (log or normal) and legend entries,
- acquire all plot coordinates

and write an equivalent `.pgf` file which typesets the plot with Pgfplots.

The indention is *not* to simulate matlab. It is a first step for a conversion. Type

```
> help matlab2pgfplots
```

on your matlab prompt for more information about its features and its limitations.

This script is experimental.

7.3 matlab2pgfplots.sh

A `bash`-script which simply starts matlab and runs

```
f=hgload( 'somefigure.fig' );
matlab2pgfplots( 'outputfile.pgf', 'fig', f );
```

See matlab2pgfplots.m above.

References

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