Manual for Package PGFPLOTS

Version 1.0

http://sourceforge.net/projects/pgfplots

Christian Feuersänger*
Institut für Numerische Simulation
Universität Bonn, Germany

June 11, 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, supporting line plots, piecewise constant plots, bar plots and area plots. It is based on Till Tantau's package PGF/TikZ.

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` for interested readers. However, some attention is useful for upgrades, which does especially affect style options: although `\tikzstyle` is still supported, you should replace any occurance of `\tikzstyle` with `\pgfplotsset{`⟨*style name*⟩`/.style={`⟨*key-value-list*⟩`}}` or the associated `.append style` variant. See section 7.10 for more detail.

3 Acknowledgements

I thank God for all hours of enjoyed programming. I thank Pascal Wolkotte, author of package `pgfgraph`, who joined development and contributed code for axis lines and tick alignment. Furthermore, I thank Dr. Schweitzer for many fruitful discussions and Dr. Meine for his ideas and suggestions.

Last but not least, I thank Till Tantau and Mark Wibrow for their excellent graphics (and more) package PGF and Ti*k*Z which is the base of PGFPLOTS.

4 Installation

4.1 Licensing

4.2 Prerequisites

PGFPLOTS requires PGF with **at least version** 2.0. It is used with

```
\usepackage{pgfplots}
```

in your preamble (see section 5.1 for information about how to use it with ConTEXt and plain TEX). There are several ways how to teach TEX where to find the files. Choose the option which fits your needs best.

4.3 Installation in Windows

Windows users often use MikTEX which downloads the latest stable package versions automatically. As far as I know, you do not need to install anything manually here.

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

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

Furthermore, you should set `TEXDOCS` as well,

```
TEXDOCS=/foo/bar/pgfplots//:
```

so that the TEX-documentation system finds the files `pgfplots.pdf` and `pgfplotstable.pdf` (on some systems, it is then enough to use `texdoc pgfplots`).

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

4.5 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`.

4.6 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.

4.7 If everything else fails...

If TEX 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.

4.8 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.

5 Drawing axes and plots

5.1 TEX-dialects: LATEX, ConTEXt, plain TEX

PGFPLOTS is compatible with LATEX, ConTEXt and plain TEX. The only difference is how to specify environments. This affects any PGF/Ti*k*Z-environments and all PGFPLOTS-environments like axis, semilogxaxis, semilogyaxis and loglogaxis:

LATEX: `\usepackage{pgfplots}` and

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

A small LATEX–example file can be found in

`doc/latex/pgfplots/pgfplotsexample.tex`.

ConTEXt: `\usemodule[pgfplots]` and

```
\starttikzpicture       \starttikzpicture
\startaxis              \startsemilogxaxis
...                     ...
\stopaxis               \stopsemilogxaxis
\stoptikzpicture        \stoptikzpicture
```

A small ConTEXt–example file can be found in
`doc/context/pgfplots/pgfplotsexample.tex`.

plain TEX: `\input pgfplots.tex` and

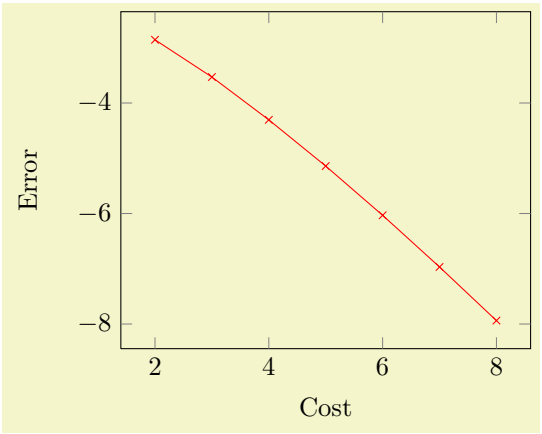
```
\tikzpicture            \tikzpicture
\axis                   \semilogxaxis
...                     ...
\endaxis                \endsemilogxaxis
\endtikzpicture         \endtikzpicture
```

A small plain–TEX–example file can be found in
`doc/plain/pgfplots/pgfplotsexample.tex`.

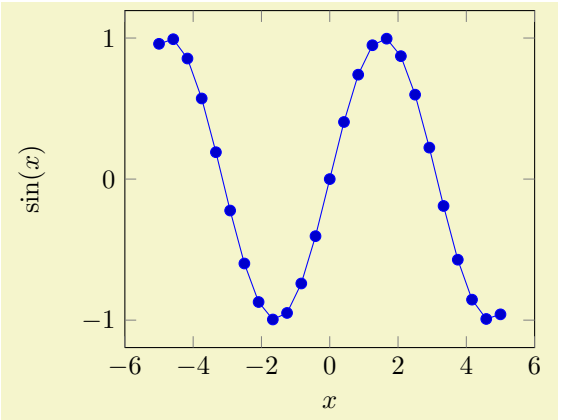
You may need to set low–level drivers if you intend to use `dvipdfm`, see section 4.8.

5.2 A first plot

Plotting is done using `\begin{axis} ... \addplot ...; \end{axis}`, where `\addplot` is an interface to the Ti*k*Z `plot` commands.




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

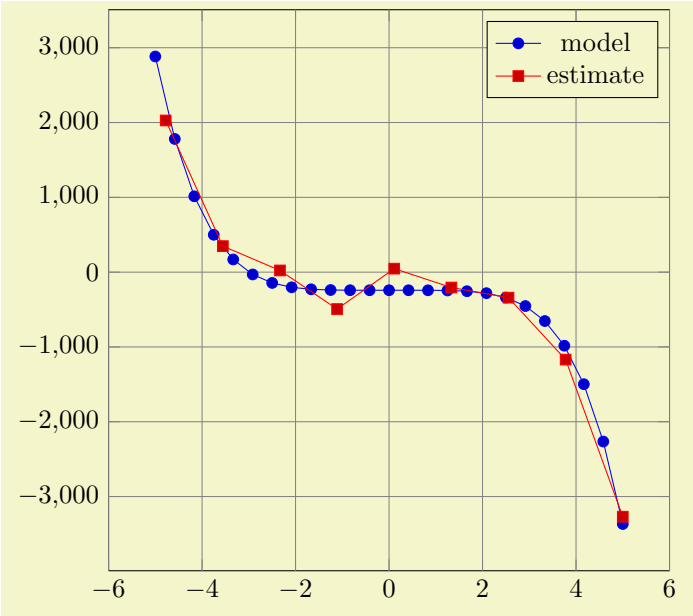



```
\begin{tikzpicture}%
    \begin{axis}[
        xlabel=$x$,
        ylabel=$\sin(x)$,
        name=an axis,
    ]
    \addplot plot[id=sin] function{sin(x)};
    \end{axis}
\end{tikzpicture}%
```

The `plot coordinates` and `plot function` commands are two of the several Ti*k*Z ways to create plots, see section 6.2 for more details[1]. The options '`xlabel`' and '`ylabel`' define axis descriptions.

5.3 Two plots in the same axis

Multiple `\addplot`-commands can be placed into the same axis.



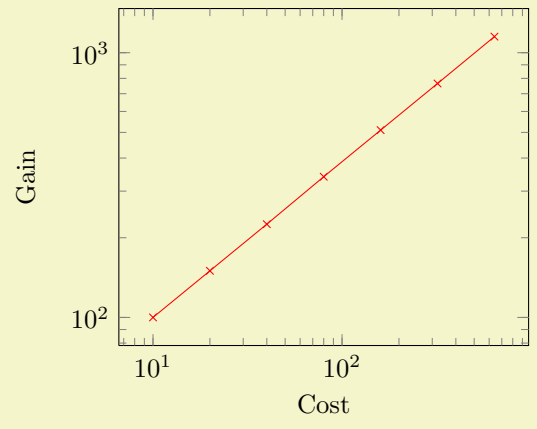

```
\begin{tikzpicture}
    \begin{axis}[
        height=9cm,
        width=9cm,
        grid=major,
    ]

    \addplot plot[id=filesuffix] function{(-x**5 - 242)};
    \addlegendentry{model}

    \addplot coordinates {
        (-4.77778,2027.60977)
        (-3.55556,347.84069)
        (-2.33333,22.58953)
        (-1.11111,-493.50066)
        (0.11111,46.66082)
        (1.33333,-205.56286)
        (2.55556,-341.40638)
        (3.77778,-1169.24780)
        (5.00000,-3269.56775)
    };
    \addlegendentry{estimate}
    \end{axis}
\end{tikzpicture}
```

A legend entry is generated if there are `\addlegendentry` commands (or one `\legend` command).

5.4 Logarithmic plots

Logarithmic plots show $\log x$ versus $\log y$ (or just one logarithmic axis). 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.

[1] Please note that you need `gnuplot` installed to use `plot function`.

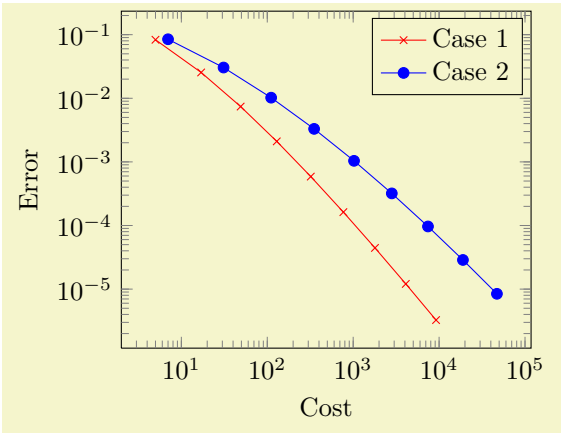



```
\begin{tikzpicture}
\begin{loglogaxis}[xlabel=Cost,ylabel=Gain]
\addplot[color=red,mark=x] coordinates {
    (10,100)
    (20,150)
    (40,225)
    (80,340)
    (160,510)
    (320,765)
    (640,1150)
};
\end{loglogaxis}
\end{tikzpicture}
```

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. Suppose we have a numeric table named `pgfplots.testtable`, containing

```
Level Cost  Error
1     7     8.471e-02
2     31    3.044e-02
3     111   1.022e-02
4     351   3.303e-03
5     1023  1.038e-03
6     2815  3.196e-04
7     7423  9.657e-05
8     18943 2.873e-05
9     47103 8.437e-06
```

then we can plot `Cost` versus `Error` using




```
\begin{tikzpicture}
\begin{loglogaxis}[
    xlabel=Cost,
    ylabel=Error]
\addplot[color=red,mark=x] 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=*]
    table[x=Cost,y=Error] {pgfplots.testtable};

\legend{Case 1,Case 2}
\end{loglogaxis}
\end{tikzpicture}
```

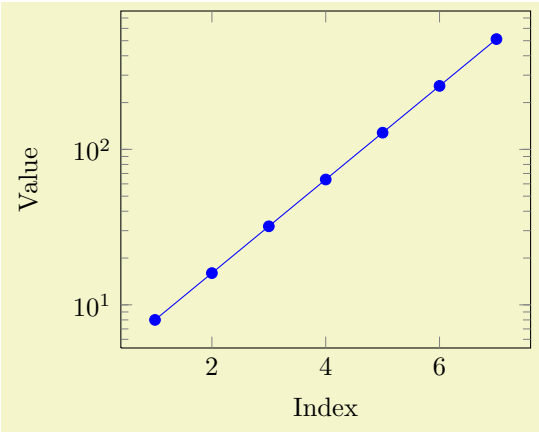
The first plot employs inline coordinates; the second one reads numerical data from file and plots column ‘`Cost`’ versus ‘`Error`’.
Besided the environment “`loglogaxis`” you can use

- `\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}
```

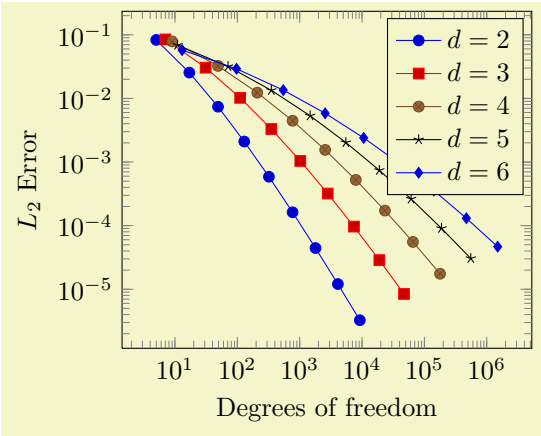
which is the same as `\begin{semilogyaxis}...\end{semilogyaxis}`.




```
\begin{tikzpicture}
    \begin{semilogyaxis}[
        xlabel=Index,ylabel=Value]

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

5.5 Cycling line styles

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

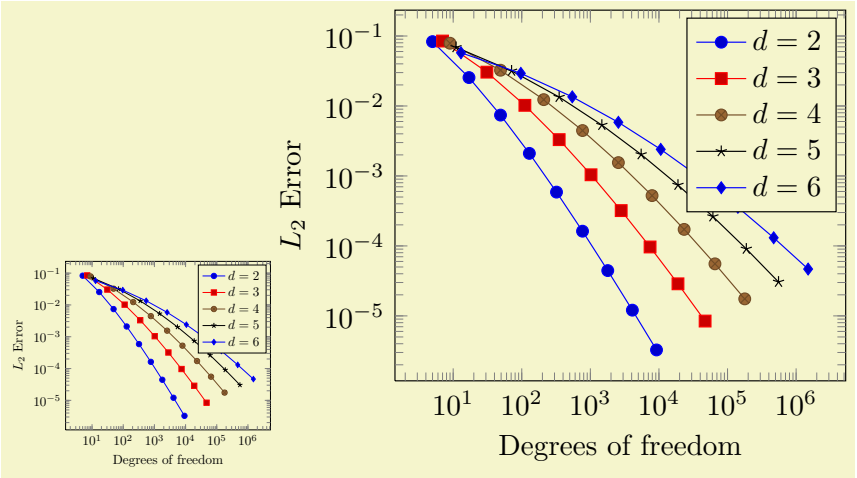


```
\begin{tikzpicture}
\begin{loglogaxis}[
    xlabel={Degrees of freedom},
    ylabel={$L_2$ Error}
]
\addplot coordinates {
    (5,8.312e-02)     (17,2.547e-02)    (49,7.407e-03)
    (129,2.102e-03)   (321,5.874e-04)   (769,1.623e-04)
    (1793,4.442e-05)  (4097,1.207e-05)  (9217,3.261e-06)
};

\addplot 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 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 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 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 cycle list can be modified, see the reference below.

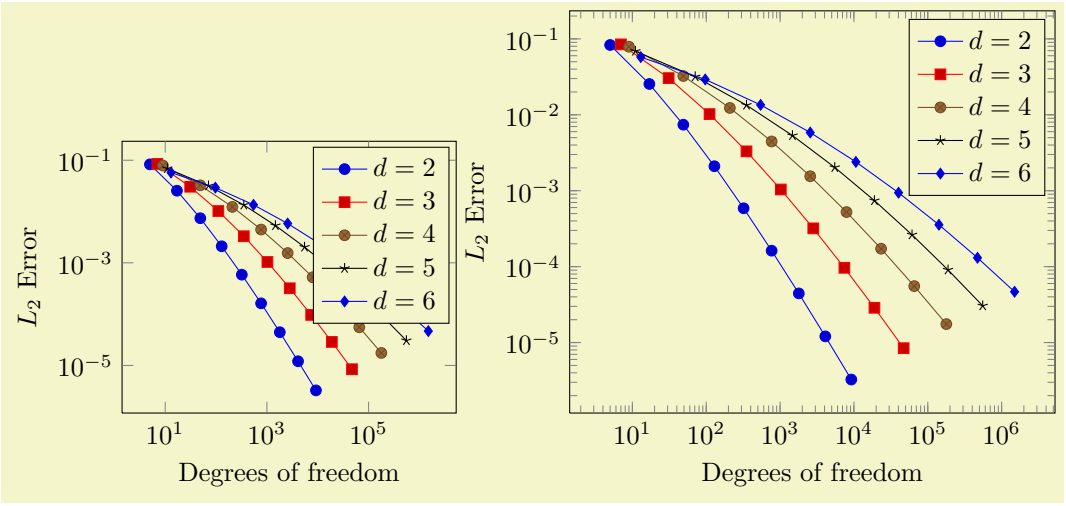
5.6 Scaling plots

You can use any of the Ti*k*Z options to modify the appearance. For example, the "`scale`" transformation takes the picture as such and scales it.



```
\begin{tikzpicture}[scale=0.5]
    \begin{loglogaxis}[
        xlabel={Degrees of freedom},
        ylabel={$L_2$ Error}
    ]
    \plotcoords
    \legend{$d=2$,$d=3$,$d=4$,$d=5$,$d=6$}
    \end{loglogaxis}
\end{tikzpicture}

\begin{tikzpicture}[scale=1.1]
    \begin{loglogaxis}[
        xlabel={Degrees of freedom},
        ylabel={$L_2$ Error}
    ]
    \plotcoords
    \legend{$d=2$,$d=3$,$d=4$,$d=5$,$d=6$}
    \end{loglogaxis}
\end{tikzpicture}
```

However, you can also scale plots by assigning a `width=5cm` and/or `height=3cm` argument. This only affects the distance of point coordinates, no font sizes or axis descriptions:




```
\begin{tikzpicture}
    \begin{loglogaxis}[
        width=6cm,
        xlabel={Degrees of freedom},
        ylabel={$L_2$ Error}
    ]
    \plotcoords
    \legend{$d=2$,$d=3$,$d=4$,$d=5$,$d=6$}
    \end{loglogaxis}
\end{tikzpicture}

\begin{tikzpicture}
    \begin{loglogaxis}[
        width=8cm,
        xlabel={Degrees of freedom},
        ylabel={$L_2$ Error}
    ]
    \plotcoords
    \legend{$d=2$,$d=3$,$d=4$,$d=5$,$d=6$}
    \end{loglogaxis}
\end{tikzpicture}
```

6 Command Reference

6.1 The Axis-environments

There is an axis environment for linear scaling, two for semi-logarithmic scaling and one for double-logarithmic scaling.

\begin{axis}{⟨*options*⟩}
  ⟨*environment contents*⟩
\end{axis}

The axis environment for normal plots with linear axis scaling.

The ‘every linear axis’ style key can be modified with

```
\pgfplotsset{every linear axis/.append style={...}}
```

to install styles specifically for linear axes. These styles can contain both Ti*k*Z- and PGFPLOTS options.

\begin{semilogxaxis}{⟨*options*⟩}
  ⟨*environment contents*⟩
\end{semilogxaxis}

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

```
\pgfplotsset{every semilogx axis/.append style={...}}
```

to install styles specifically for the case with `xmode=log`, `ymode=normal`.

\begin{semilogyaxis}{⟨*options*⟩}
  ⟨*environment contents*⟩
\end{semilogyaxis}

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

The style ‘every semilogy axis’ will be installed for each such plot.

\begin{loglogaxis}{⟨*options*⟩}
  ⟨*environment contents*⟩
\end{loglogaxis}

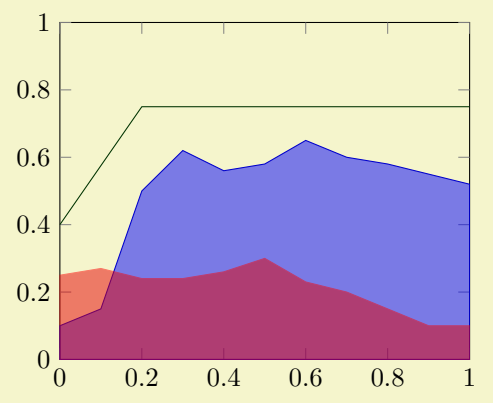
The axis environment for logarithmic scaling of both, x and y axes, As for the other axis possibilities, there is a style ‘every loglog axis’ which is installed at the environment’s beginning.

They are all equivalent to

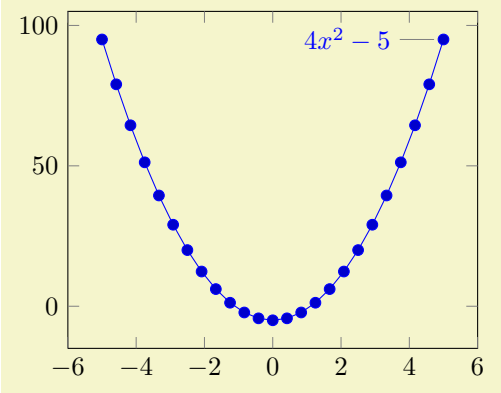
```
\begin{axis}[
    xmode=log|normal,
    ymode=log|normal]
...
\end{axis}
```

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

6.2 The Plot Command




```
\begin{tikzpicture}
\begin{axis}[ymin=0,ymax=1,enlargelimits=false]
\addplot
    [blue!80!black,fill=blue,fill opacity=0.5]
coordinates
{(0,0.1)    (0.1,0.15)  (0.2,0.5)   (0.3,0.62)
 (0.4,0.56) (0.5,0.58)  (0.6,0.65)  (0.7,0.6)
 (0.8,0.58) (0.9,0.55)  (1,0.52)}
|- (axis cs:0,0) -- cycle;

\addplot
    [red,fill=red!90!black,opacity=0.5]
coordinates
{(0,0.25)   (0.1,0.27)  (0.2,0.24)  (0.3,0.24)
 (0.4,0.26) (0.5,0.3)   (0.6,0.23)  (0.7,0.2)
 (0.8,0.15) (0.9,0.1)   (1,0.1)}
|- (axis cs:0,0) -- cycle;

\addplot[green!20!black] coordinates
    {(0,0.4) (0.2,0.75) (1,0.75)};
\end{axis}
\end{tikzpicture}
```




```
\begin{tikzpicture}
\begin{axis}
\addplot plot
    [id=parable,domain=-5:5]
    function{4*x**2 - 5}
    node[pin=180:{$4x^2-5$}]{};
\end{axis}
\end{tikzpicture}
```

\addplot[⟨*style options*⟩] plot[⟨*behavior options*⟩] ⟨*input data*⟩ ⟨*trailing path commands*⟩;

This is the main plotting command, available within each axis environment.

It reads point coordinates from one of the available input sources specified by ⟨*input data*⟩, updates limits, remembers ⟨*style options*⟩ for use in a legend (if any) and applies any necessary coordinate transformations (or logarithms).

The ⟨*style options*⟩ can be omitted in which case the next entry from the `cycle list` will be inserted as ⟨*style options*⟩. These keys characterize the plot's type like linear interpolation, smooth plot, constant interpolation or bar plot and define colors, markers and line specifications.

The optional ⟨*behavior options*⟩ can be used to modify plot variants, for example to add error bars. They are described when needed.

The ⟨*input data*⟩ is one of several coordinate input tools which are described in more detail below. Finally, if `\addplot` successfully processed all coordinates from ⟨*input data*⟩, it generates Ti*k*Z-drawing commands (for example `plot coordinate {...}`). If ⟨*trailing path commands*⟩ is not empty, these arguments are appended to the final drawing command.

Some more details:

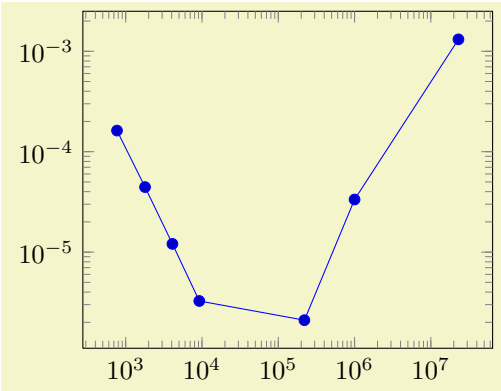
- The style `/pgfplots/every axis plot` will be installed at the beginning of ⟨*style options*⟩. That means you can use

  ```
  \pgfplotsset{every axis plot/.append style={...}}
  ```

  to add options to all your plots - maybe to set line widths to `thick`. Furthermore, if you have more than one plot inside of an axis, you can also use

  ```
  \pgfplotsset{every axis plot no 3/.append style={...}}
  ```

  to modify options for the plot with number 3 only. The first plot has number 0.
- The ⟨*style options*⟩ are remembered for the legend. Furthermore, they are available as '`current plot style`' as long as the path is not yet finished or in associated error bars.
- See subsection 7.3 for a list of available markers and line styles.
- For log plots, PGFPLOTS will compute the natural logarithm log(·) 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 coordinates {
    (769,     1.6227e-04)
    (1793,     4.4425e-05)
    (4097,     1.2071e-05)
    (9217,     3.2610e-06)
    (2.2e5,     2.1E-6)
    (1e6,     0.00003341)
    (2.3e7,     0.00131415)
};
\end{loglogaxis}
\end{tikzpicture}
```

You can represent arbitrarily 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 applies floating point arithmetics to support large or small numbers like 0.00000001234 or $1.234 \cdot 10^{24}$. Its number range is much larger than TEX's native support for numbers. The relative precision is at least 5 significant decimal digits for the mantisse. As soon as the axes limits are completely known, PGFPLOTS applies a transformation which maps these floating point numbers into TEX-precision using transformations

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

with properly chosen integers $s_x, s_y \in \mathbb{Z}$ and shifts $a_x, a_y \in \mathbb{R}$. Section 7.12 contains a description of `disabledatascaling` and provides more details about the transformation.

- 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. Every coordinate will be checked. Care has been taken to avoid TEX's limited numerical capabilities.
  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}`.

6.2.1 Providing Input Coordinates

`\addplot coordinates {`⟨*coordinate list*⟩`};`
`\addplot[`⟨*style options*⟩`] plot[`⟨*behavior options*⟩`] coordinates {`⟨*coordinate list*⟩`}` ⟨*trailing path commands*⟩`;`

The '`plot coordinates`' command is provided by Ti*k*Z and reads its input data from a sequence of point coordinates.

```
\addplot plot coordinates {
    (0,0)
    (0.5,1)
    (1,2)
};
```

You can also supply error coordinates (reliability bounds) if you are interested in error bars. Simply append the error coordinates with '`+- ({`⟨*ex*⟩`},{`⟨*ey*⟩`})`' to the associated coordinate:

```
\addplot plot coordinates {
    (0,0)   +- (0.1,0)
    (0.5,1) +- (0.4,0.2)
    (1,2)
    (2,5)   +- (1,0.1)
};
```

or

```
\addplot plot coordinates {
    (1300,1e-6) +- (0.1,0.2)
    (2600,5e-7) +- (0.2,0.5)
    (4000,1e-7) +- (0.1,0.01)
};
```

These error coordinates are only used in case of error bars, see section 7.6. You will also need to configure whether these values denote absolute or relative errors.

The coordinates as such can be numbers as `+5`, `-1.2345e3`, `35.0e2`, `0.00000123` or `1e2345e-8`. They are not limited to TEX's precision.

`\addplot file {`⟨*name*⟩`};`

`\addplot[⟨style options⟩] plot[⟨behavior options⟩] file {⟨name⟩} ⟨trailing path commands⟩;`

PGFPLOTS supports two ways to read plot coordinates of external files, and one of them is the TikZ-command '`plot file`'. It is to be used like

```
\addplot plot file {datafile.dat};
```

where {⟨*name*⟩} is a text file with at least two columns which will be used as x and y coordinates. Such files are often generated by GNUPLOT:

```
#Curve 0, 20 points
#x y type
0.00000 0.00000 i
0.52632 0.50235 i
1.05263 0.86873 i
1.57895 0.99997 i
...
9.47368 -0.04889 i
10.00000 -0.54402 i
```

This listing has been copied from [1, section 16.4]. The mentioned section also contains some more details about using `plot file`.

`\addplot table [⟨column selection⟩]{⟨file⟩};`
`\addplot[⟨style options⟩] plot[⟨behavior options⟩] table [⟨column selection⟩]{⟨file⟩} ⟨trailing path commands⟩;`

PGFPLOTS comes with a new plotting command, the '`plot table`' macro. It's usage is similar in spirit to '`plot file`', but its flexibility is higher. Given a data file like

```
dof     L2              Lmax            maxlevel
5       8.31160034e-02  1.80007647e-01  2
17      2.54685628e-02  3.75580565e-02  3
49      7.40715288e-03  1.49212716e-02  4
129     2.10192154e-03  4.23330523e-03  5
321     5.87352989e-04  1.30668515e-03  6
769     1.62269942e-04  3.88658098e-04  7
1793    4.44248889e-05  1.12651668e-04  8
4097    1.20714122e-05  3.20339285e-05  9
9217    3.26101452e-06  8.97617707e-06  10
```

one may want to plot '`dof`' versus '`L2`' or '`dof`' versus '`Lmax`'. This can be done by

```
\begin{tikzpicture}
\begin{loglogaxis}[
    xlabel=Dof,
    ylabel=$L_2$ error$]
\addplot table[x=dof,y=L2] {datafile.dat};
\end{loglogaxis}
\end{tikzpicture}
```

or

```
\begin{tikzpicture}
\begin{loglogaxis}[
    xlabel=Dof,
    ylabel=$L_infty$ error$]
\addplot table[x=dof,y=Lmax] {datafile.dat};
\end{loglogaxis}
\end{tikzpicture}
```

Alternatively, you can load the table *once* and use it *multiple* times:

```
\pgfplotstableread{datafile.dat}\table
...
\addplot table[x=dof,y=L2] from \table;
...
\addplot table[x=dof,y=Lmax] from \table;
...
```

I am not really sure how much time can be saved, but it works anyway.

If you do prefer to access columns by column indices instead of column names (or your tables do not have column names), you can also use

```
\addplot table[x index=2,y index=3] {datafile.dat};
\addplot table[x=dof,y index=2] {datafile.dat};
```

Summary and remarks:

- Use `plot table[x={`⟨*column name*⟩`},y={`⟨*column name*⟩`}]` to access column names. Those names are case sensitive and need to exist.
- Use `plot table[x index={`⟨*column index*⟩`},y index={`⟨*column index*⟩`}]` to access column indices. Indexing starts with 0. You may also use an index for x and a column name for y.
- Use `plot table[header=false] {`⟨*file name*⟩`}` if your input file has no column names. Otherwise, the first non-comment line is checked for column names: if all entries are numbers, they are treated as numerical data; if one of them is not a number, all are treated as column names.
- It is possible to read error coordinates from tables as well. Simply add options '`x error`', '`y error`' or '`x error index`'/'`y error index`' to `{`⟨*source columns*⟩`}`. See section 7.6 for details about error bars.
- Use `plot table[`⟨*source columns*⟩`] from {`⟨*macro*⟩`}` to use a pre–read table. Tables can be read using

  ```
  \pgfplotstableread{datafile.dat}\macroname.
  ```

- The accepted input format of those tables is as follows:
  - Columns are usually separated by white spaces (at least one tab or space).
    If you need other column separation characters, you can use the
    col sep=space|comma|colon|semicolon|braces
    option which is documented in all detail in the manual for PgfplotsTable which is part of pgfplots.
  - Any line starting with '#' or '%' is ignored.
  - The first line will be checked if it contains numerical data. If there is a column in the first line which is *no* number, the complete line is considered to be a header which contains column names. Otherwise it belongs to the numerical data and you need to access column indices instead of names.
  - There is future support for a second header line which must start with '`$flags `'. Currently, such a line is ignored. It may be used to provide number formatting hints like precision and number format if those tables shall be typeset using `\pgfplotstabletypeset` (see the manual for PgfplotsTable).
  - The accepted number format is the same as for '`plot coordinates`', see above.
  - If you omit column selectors, the default is to plot the first column against the second. That means `plot table` does exactly the same job as `plot file` for this case.

`\addplot function {`⟨*gnuplot code*⟩`};`
`\addplot[`⟨*style options*⟩`] plot[`⟨*behavior options*⟩`] function {`⟨*gnuplot code*⟩`}` ⟨*trailing path commands*⟩`;`

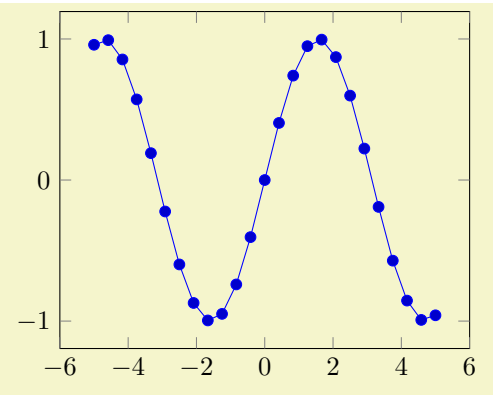
The plot function command uses the external program `gnuplot` to compute coordinates. The resulting coordinates are written to a text file which will be plotted with `plot file`. pgf checks whether coordinates need to be re-generated and calls `gnuplot` whenever necessary (this is usually the case if you change the number of samples, the argument to `plot function` or the plotted domain[2]).

Since system calls are a potential danger, they need to be enabled explicitly using command line options, for example

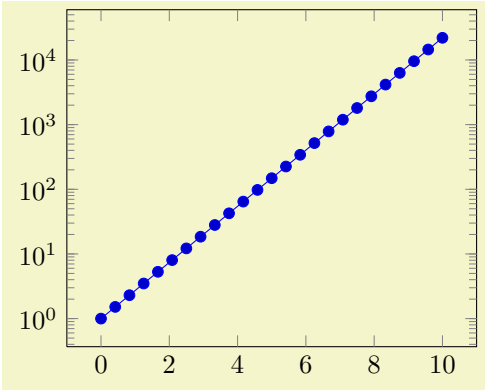
```
pdflatex -shell-escape filename.tex.
```

Sometimes it is called `shell-escape` or `enable-write18`.

[2]Please note that pgfplots produces slightly different files than Ti*k*Z when used with `plot function` (it configures high precision output). You should use different ids to avoid conflicts in such a case.




```
\begin{tikzpicture}
\begin{axis}
\addplot plot[id=sin]
    function{sin(x)};
\end{axis}
\end{tikzpicture}
```




```
\begin{tikzpicture}
\begin{semilogyaxis}
\addplot plot[id=exp,domain=0:10]
    function{exp(x)};
\end{semilogyaxis}
\end{tikzpicture}
```

The ⟨*style options*⟩ determine the appearance of the plotted function; these parameters also affect the legend. The ⟨*behavior options*⟩ are specific to the gnuplot interface. These options are described in all detail in [1, section 18.6]. A short summary is shown below.

`/tikz/domain`=[⟨*start*⟩:⟨*end*⟩] (no default, initially `[-5:5]`)

Determines the plotted range. This is not necessarily the same as the axis limits (which are configured with the `xmin`/`xmax` options).

This option is also used for plot expression, see below.

`/tikz/id`={⟨*unique string identifier*⟩} (no default)

A unique identifier for the current plot. It is used to generate temporary filenames for `gnuplot` output.

`/tikz/prefix`={⟨*file name prefix*⟩} (no default)

A common path prefix for temporary filenames (see [1, section 18.6] for details).

`/tikz/samples`={⟨*number*⟩} (no default)

Sets the number of sample points.

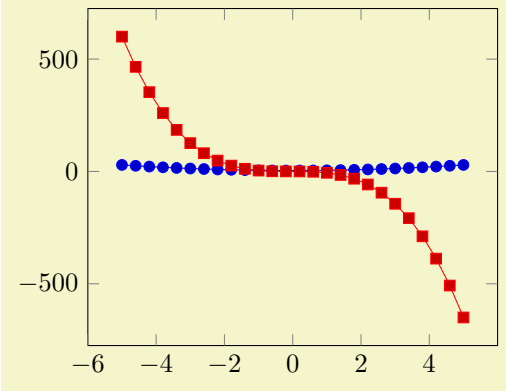
`/tikz/raw gnuplot` (no value)

Disables the use of `samples` and `domain`.

Please refer to [1, section 18.6] for more details about `plot function` and the `gnuplot` interaction.

`\addplot` (⟨*x expression*⟩,⟨*y expression*⟩) ;
`\addplot[`⟨*style options*⟩`] plot[`⟨*behavior options*⟩`]` (⟨*x expression*⟩,⟨*y expression*⟩) ⟨*trailing path commands*⟩;

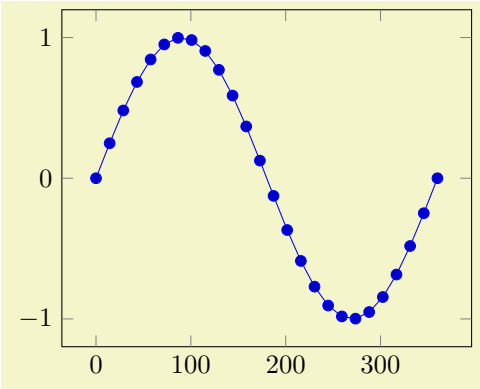
Similar to `plot function`, this input method allows to provide expressions which will be sampled. But unlike `plot function`, the expressions are evaluated using the math parser of PGF, no external program is required.




```
\begin{tikzpicture}
\begin{axis}
    \addplot (\x,\x^2 + 4);
    \addplot (\x,-5*\x^3 - \x^2);
\end{axis}
\end{tikzpicture}
```

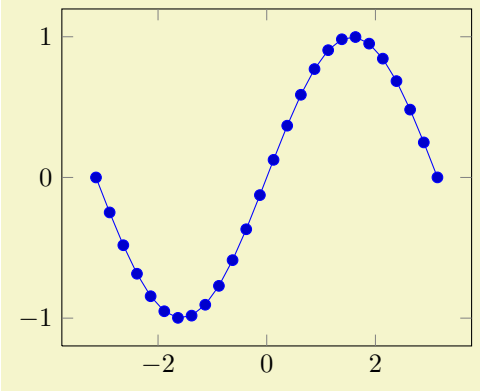
The number of points and the sampled range is configured as for `plot function`, that means using `samples` and `domain`.

Please note that PGF's math parser uses degrees for trigonometric functions:




```
\begin{tikzpicture}
\begin{axis}
    \addplot plot[domain=0:360]
        (\x,{sin(\x)});
\end{axis}
\end{tikzpicture}
```

the braces are necessary to delimit the y argument here. If you want to use radians, use

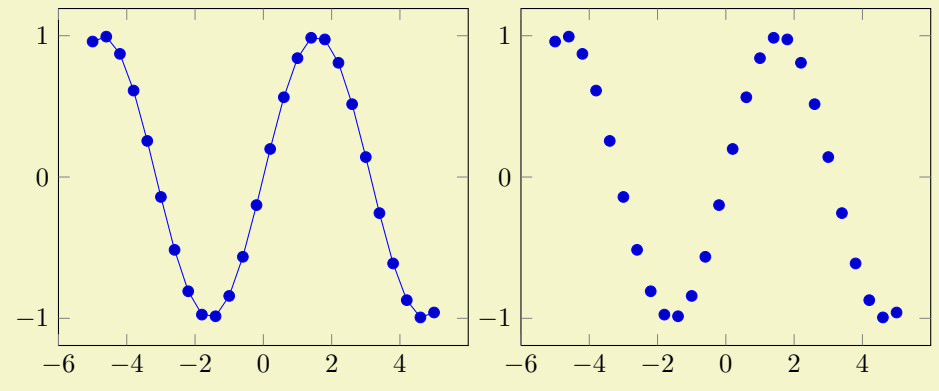



```
\begin{tikzpicture}
\begin{axis}
    \addplot plot[domain=-3.14159:3.14159]
        (\x,{sin(\x r)});
\end{axis}
\end{tikzpicture}
```

to convert the radians to degrees. The Ti*k*Z plot expression parser also accepts some more options like `samples at={`⟨*coordinate list*⟩`}` or `variable=\t` which are described in the Ti*k*Z manual.

`\addplot+`[⟨*style options*⟩] `...;`

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




```
\begin{tikzpicture}
\begin{axis}
\addplot (\x,{sin(\x r)});
\end{axis}
\end{tikzpicture}

\begin{tikzpicture}
\begin{axis}
\addplot+[only marks] (\x,{sin(\x r)});
\end{axis}
\end{tikzpicture}
```

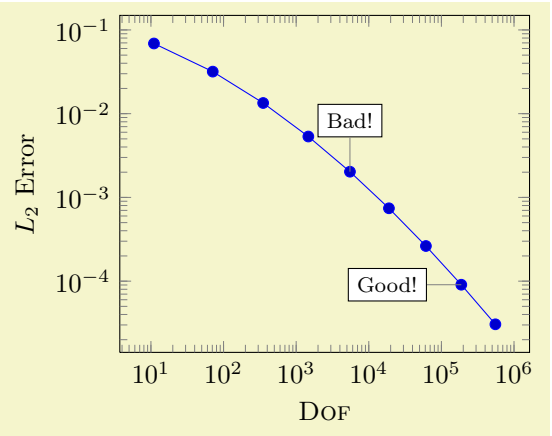
6.3 Accessing Axis Coordinates for Annotations

Coordinate system `axis cs`

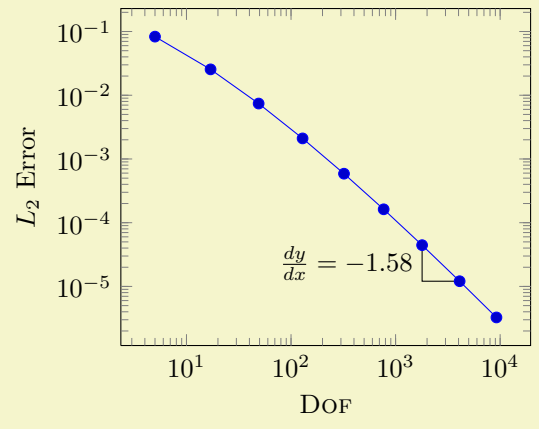
PGFPLOTS provides a new coordinate system for use inside of an axis, the "axis coordinate system", `axis cs`.

It can be used to draw any Ti*k*Z-graphics at axis coordinates. It is used like

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




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

    \addplot 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)
    };

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




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

\addplot 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}
```

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

Predefined node `current plot begin`

This coordinate will be defined for every plot and can be used is ⟨*trailing path commands*⟩ or after a plot. It is the first coordinate of the current plot.

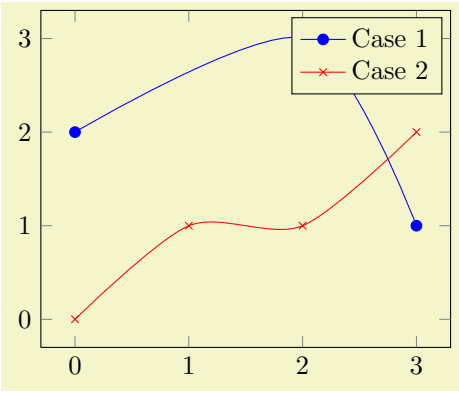
Predefined node `current plot end`

This coordinate will be defined for every plot. It is the last coordinate of the current plot.

6.4 Legend Commands

`\addlegendentry{`⟨*name*⟩`}`

Adds a single legend entry to the legend list. This will also enable legend drawing.




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

\addplot[smooth,color=red,mark=x]
    coordinates {
        (0,0)
        (1,1)
        (2,1)
        (3,2)
    };
\addlegendentry{Case 2}
\end{axis}
\end{tikzpicture}
```

It does not matter where `\addlegendentry` commands are placed, only the sequence matters. You will need one `\addlegendentry` for every `\addplot` command.

Optional argument are accepted with`\addlegendentry[`⟨*key-value-list*⟩`]{...}`. This does mainly affect some keys affecting the legend layout, support is very limited.

`\legend{⟨list⟩}`

You can use `\legend{⟨list⟩}` to assign a complete legend.

```
\legend{$d=2$,$d=3$,$d=4$,$d=5$,$d=6$}
```

The argument of `\legend` is a comma–separated list of entries, one for each plot. It is processed using the PGF-foreach command[3]. The short marker/line combination shown in legends is acquired from the `{⟨style options⟩}` argument of `\addplot`.

6.4.1 Legend Appearance

`/pgfplots/every axis legend` (style, no value)

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

```
\pgfplotsset{every axis legend/.append style={
        at={(0,0)},
        anchor=south west}}
```

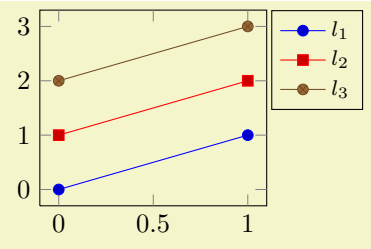
will draw it at the lower left corner of the axis while

```
\pgfplotsset{every axis legend/.append style={
        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 $(1,1)$.

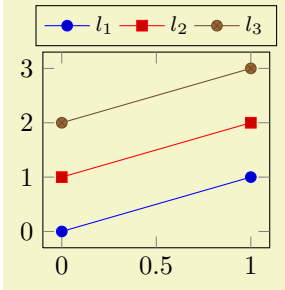
The legend is a Ti*k*Z-matrix, so you can use any Ti*k*Z option which affects nodes and matrizes (see [1, section 13 and 14]). The matrix is created by something like

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




```
\pgfplotsset{every axis legend/.append style={
        at={(1.02,1)},
        anchor=north west}}
\begin{tikzpicture}
\begin{axis}
\addplot coordinates {(0,0) (1,1)};
\addplot coordinates {(0,1) (1,2)};
\addplot coordinates {(0,2) (1,3)};
\legend{$l_1$,$l_2$,$l_3$}
\end{axis}
\end{tikzpicture}
```

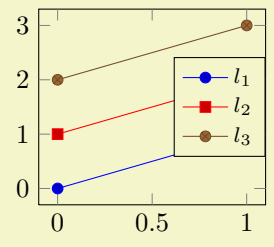
Use `legend columns={⟨number⟩}` to configure the number of horizontal legend entries.




```
\begin{tikzpicture}
\pgfplotsset{every axis legend/.append style={
        at={(0.5,1.03)},
        anchor=south}}
\begin{axis}[legend columns=4]
\addplot coordinates {(0,0) (1,1)};
\addplot coordinates {(0,1) (1,2)};
\addplot coordinates {(0,2) (1,3)};
\legend{$l_1$,$l_2$,$l_3$}
\end{axis}
\end{tikzpicture}
```

Instead of the `.append style`, you can also use `legend style` as in the following example. It has the same effect.

[3] Older versions of PGFPLOTS used `\legend{first\\second\\third\\}` instead of comma–separated lists. This syntax is still accepted.




```
\begin{tikzpicture}
\begin{axis}[
    legend style={
        at={(1,0.5)},
        anchor=east}]
\addplot coordinates {(0,0) (1,1)};
\addplot coordinates {(0,1) (1,2)};
\addplot coordinates {(0,2) (1,3)};
\legend{$l_1$,$l_2$,$l_3$}
\end{axis}
\end{tikzpicture}
```

The default `every axis legend` style is

```
\pgfplotsset{every axis legend/.style={%
    cells={anchor=center},
    inner xsep=3pt,inner ysep=2pt,nodes={inner sep=2pt,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

```
\pgfplotsset{every axis legend/.append style={...}}
```

`\autoplotspeclist`

This command should no longer be used, although it will be kept as technical implementation detail. Please use the '`cycle list`' option, section 7.3.4.

`\pgfmathlogtologten`⟨*number*⟩

Assigns the result of ⟨*number*⟩/ log(10) to `\pgfmathresult`.

`\logten`

Expands to the constant log(10). Useful for logplots because $\log(10^i) = i \log(10)$. This command is only available inside of an Ti*k*Z-picture.

`\pgfmathprintnumber{`⟨*number*⟩`}`

Generates pretty–printed output[4] for `{`⟨*number*⟩`}`. This method is used for every tick label.

The number is printed using the current number printing options, see section 7.7 for the different number styles, rounding precision and rounding methods.

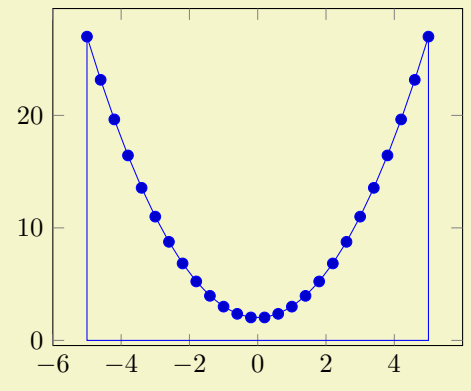
6.5 Closing Plots

`\closedcycle`

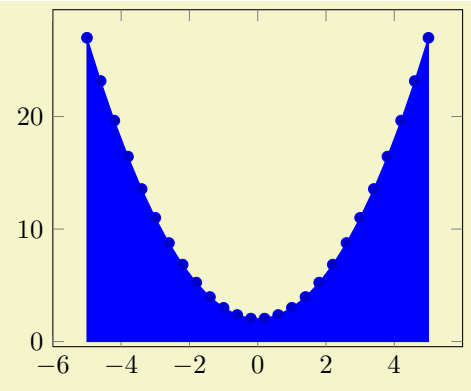
Provide `\closedcycle` as ⟨*trailing path commands*⟩ after `\addplot` to draw a closed line from the last plot coordinate to the first one.

Use `\closedcycle` whenevery you intend to fill the area under a plot.

[4] This method was previously `\prettyprintnumber`. It's functionality has been included into PGF and the old command is now deprecated.

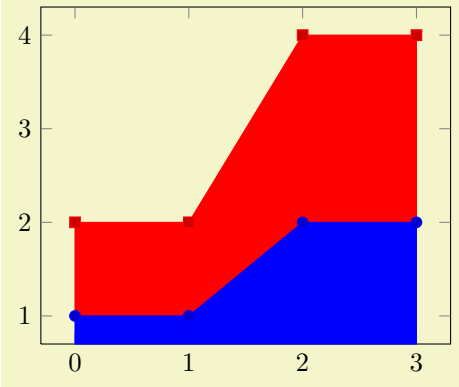



```
\begin{tikzpicture}
    \begin{axis}
    \addplot (\x,\x^2+2) \closedcycle;
    \end{axis}
\end{tikzpicture}
```




```
\begin{tikzpicture}
    \begin{axis}
    \addplot+[fill] (\x,\x^2+2) \closedcycle;
    \end{axis}
\end{tikzpicture}
```

In case of stacked plots, `\closedcycle` connects the current plot with the previous plot instead of connecting with the x axis[5].




```
\begin{tikzpicture}
    \begin{axis}[stack plots=y]
    \addplot+[fill] coordinates
        {(0,1) (1,1) (2,2) (3,2)} \closedcycle;
    \addplot+[fill] coordinates
        {(0,1) (1,1) (2,2) (3,2)} \closedcycle;
    \end{axis}
\end{tikzpicture}
```

6.6 Other Commands

`\plotnum`

Inside of `\addplot` or any associated style, option or command, `\plotnum` expands to the current plot's number, starting with 0.

`\numplots`

Inside of any of the axis environments, associated style, option or command, `\numplots` expands to the total number of plots.

`\coordindex`

[5]The implementation for stacked plots requires some additional logic to determine the filled area: `\closedcycle` will produce a `plot coordinates` command with *reversed* coordinates of the previous plot. This is usually irrelevant for end users, but it assumes that the plot's type is symmetric. Since constant plots are inherently unsymmetric, `\closedcycle` will use `const plot mark right` as reversed sequence for `const plot mark left`.

Inside of an `\addplot` command, this macro expands to the number of the actual coordinate (starting with 0).

It is useful together with `x filter` or `y filter` to (de-)select coordinates.

`\pgfplotstableread{`⟨*file*⟩`}`

Please refer to the manual of PgfplotsTable, `pgfplotstable.pdf`, which is part of the pgfplots-bundle.

`\pgfplotstabletypeset{`⟨*macro*⟩`}`

Please refer to the manual of PgfplotsTable, `pgfplotstable.pdf`, which is part of the pgfplots-bundle.

`\pgfplotstabletypesetfile{`⟨*file*⟩`}`

Please refer to the manual of PgfplotsTable, `pgfplotstable.pdf`, which is part of the pgfplots-bundle.

7 Option Reference

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

```
\pgfplotsset{every axis/.append style={line width=1pt}}
```

for example. There are several other styles predefined to modify the appearance, see section 7.10.

7.1 Pgfplots Options and Ti*k*Z Options

This section is more or less technical and can be skipped unless one really wants to know more about this topic.

Ti*k*Z options and pgfplots options can be mixed inside of the axis arguments and in any of the associated styles. For example,

```
\pgfplotsset{every axis legend/.append style={
    legend columns=3,font=\Large}}
```

assigns the '`legend columns`' option (a pgfplots option) and uses '`font`' for drawing the legend (a Ti*k*Z option).

The axis environments will process any known pgfplots options, and all '`every`'–styles will be parsed for pgfplots options. Every unknown option is supposed to be a Ti*k*Z option and will be forward to the associated Ti*k*Z drawing commands. For example, the '`font=\`**`Large`**' above will be used as argument to the legend matrix, and the '`font=\`**`Large`**' argument in

```
\pgfplotsset{every axis label/.append style={
    ylabel=Error,xlabel=Dof,font=\Large}}
```

will be used in the nodes for axis labels (but not the axis title, for example).

It is an error if you assign incompatible options to axis labels, for example '`xmin`' and '`xmax`' can't be set inside of '`every axis label`'.

7.2 Plot Types

pgfplots supports several two-dimensional line-plots like piecewise linear line plots, piecewise constant plots, smoothed plots, bar plots and comb plots. Most of them use the pgf plot handler library directly, see [1, section 18.8].

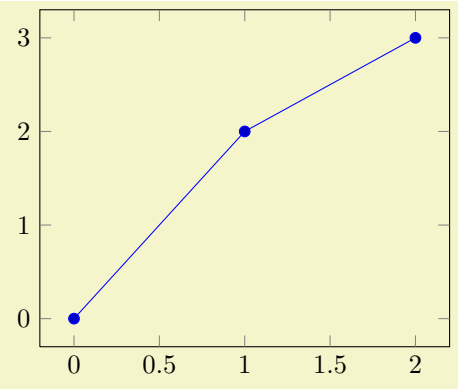
Plot types are part of the plot style, so they are set with options. The following list contains a short summary of the pgf plot library, [1, section 18.8].

7.2.1 Linear Plots

/tikz/sharp plot (no value)

\addplot+[sharp plot]

Linear ('sharp') plots are the default. Point coordinates are simply connected by straight lines.




```
\begin{tikzpicture}
\begin{axis}
    \addplot+[sharp plot] coordinates
        {(0,0) (1,2) (2,3)};
\end{axis}
\end{tikzpicture}
```

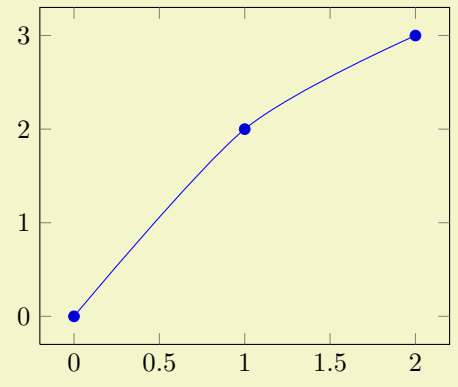
The '+' here means to use the normal plot cycle list and append '`sharp plot`' to its option list.

7.2.2 Smooth Plots

/tikz/smooth (no value)

\addplot+[smooth]

Smooth plots interpolate smoothly between successive points.




```
\begin{tikzpicture}
\begin{axis}
    \addplot+[smooth] coordinates
        {(0,0) (1,2) (2,3)};
\end{axis}
\end{tikzpicture}
```

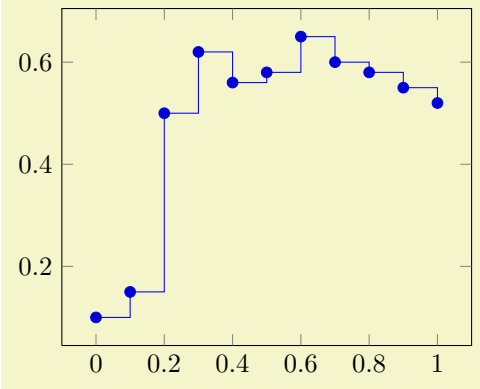
7.2.3 Constant Plots

Constant plots draw lines parallel to the x-axis to connect coordinates. The discontinuos edges may be drawn or not, and marks may be placed on left or right ends.

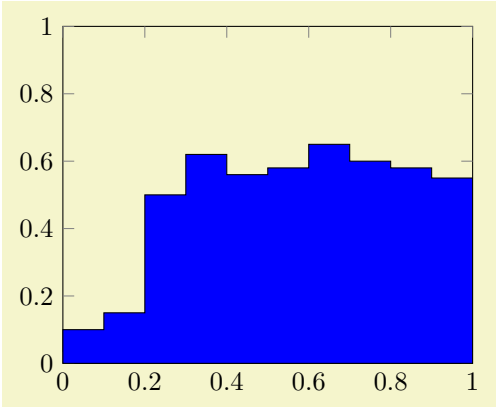
/tikz/const plot (no value)

\addplot+[const plot]

Connects all points with horizontal and vertical lines. Marks are placed left-handed on horizontal line segments, causing the plot to be right-sided continuous at all data points.




```
\begin{tikzpicture}
\begin{axis}
\addplot+[const plot]
coordinates
{(0,0.1)    (0.1,0.15)  (0.2,0.5)   (0.3,0.62)
 (0.4,0.56) (0.5,0.58)  (0.6,0.65)  (0.7,0.6)
 (0.8,0.58) (0.9,0.55)  (1,0.52)};
\end{axis}
\end{tikzpicture}
```




```
\begin{tikzpicture}
\begin{axis}[ymin=0,ymax=1,enlargelimits=false]
\addplot
    [const plot,fill=blue,draw=black]
coordinates
{(0,0.1)    (0.1,0.15)  (0.2,0.5)   (0.3,0.62)
 (0.4,0.56) (0.5,0.58)  (0.6,0.65)  (0.7,0.6)
 (0.8,0.58) (0.9,0.55)  (1,0.52)}
    \closedcycle;
\end{axis}
\end{tikzpicture}
```

/tikz/const plot mark left (no value)

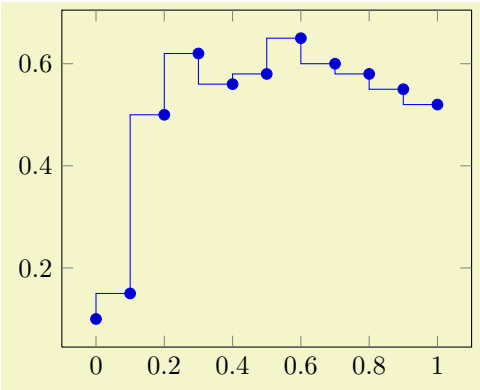
\addplot+[const plot mark left]

An alias for ‘const plot’.

/tikz/const plot mark right (no value)

\addplot+[const plot mark right]

A variant which places marks on the right of each line segment, causing plots to be left-sided continuous at coordinates.

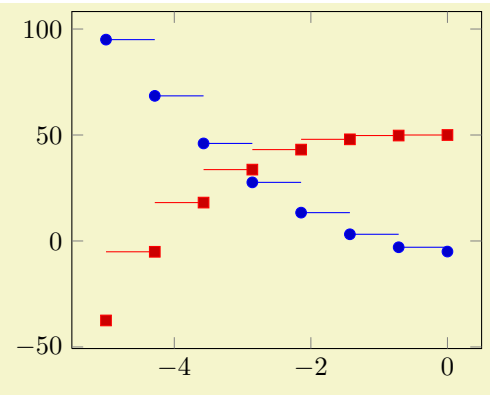



```
\begin{tikzpicture}
\begin{axis}
\addplot+[const plot mark right]
coordinates
{(0,0.1)    (0.1,0.15)  (0.2,0.5)   (0.3,0.62)
 (0.4,0.56) (0.5,0.58)  (0.6,0.65)  (0.7,0.6)
 (0.8,0.58) (0.9,0.55)  (1,0.52)};
\end{axis}
\end{tikzpicture}
```

/tikz/jump mark left (no value)

\addplot+[jump mark left]

A variant of ‘const plot mark left’ which does not draw vertical lines.




```
\begin{tikzpicture}[samples=8]
\begin{axis}
\addplot+[jump mark left]
    plot[id=parablex,domain=-5:0]
    function{4*x**2 - 5};

\addplot+[jump mark right]
    plot[id=cubic,domain=-5:0]
    function{0.7*x**3 + 50};
\end{axis}
\end{tikzpicture}
```

/tikz/jump mark right (no value)

\addplot+[jump mark right]

A variant of 'const plot mark right' which does not draw vertical lines.

7.2.4 Bar Plots

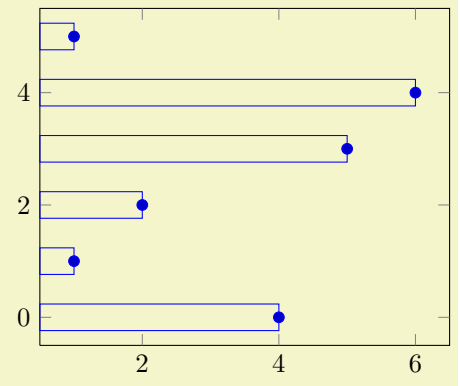
Bar plots place horizontal or vertical bars at coordinates. Multiple bar plots in one axis can be stacked on top of each other or aligned next to each other.

/tikz/xbar (no value)

\addplot+[xbar]

Places horizontal bars between the $(y = 0)$ line and each coordinate.

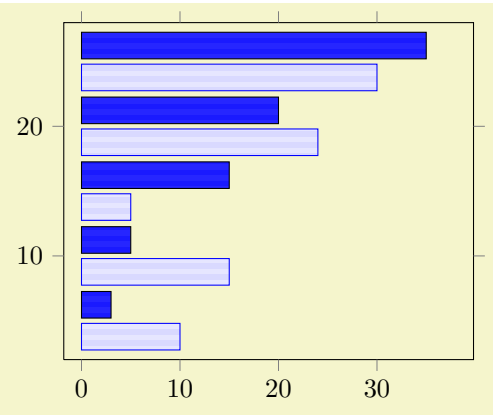
This option is used on a per-plot basis and configures only the visualization of coordinates. The figure-wide style /pgfplots/xbar also sets reasonable options for ticks, legends and multiple plots.




```
\begin{tikzpicture}
\begin{axis}
\addplot+[xbar] coordinates
    {(4,0) (1,1) (2,2)
     (5,3) (6,4) (1,5)};
\end{axis}
\end{tikzpicture}
```

Bars are centered at plot coordinates with width bar width. Using bar plots usually means more than just a different way of how to connect coordinates, for example to draw ticks outside of the axis, change the legend's appearance or introduce shifts if multiple \addplot commands appear.

There is a preconfigured style for xbar which is installed automatically if you provide xbar as argument to the axis environment which provides this functionality.




```
\begin{tikzpicture}
\begin{axis}[xbar,enlargelimits=0.15]
\addplot
[draw=blue,pattern=horizontal lines light blue]
coordinates
    {(10,5) (15,10) (5,15) (24,20) (30,25)};

\addplot
[draw=black,pattern=horizontal lines dark blue]
coordinates
    {(3,5) (5,10) (15,15) (20,20) (35,25)};
\end{axis}
\end{tikzpicture}
```

Here `xbar` yields `/pgfplots/xbar` because it is an argument to the axis, not to a single plot.

Besides line-, fill- and colorstyles, bars can be configured with `bar width` and `bar shift`, see below.

`/pgfplots/xbar={`⟨*shift for multiple plots*⟩`}` (style, default `2pt`)

This style sets `/tikz/xbar` *and* some commonly used options concerning horizontal bars for the complete axis. This is automatically done if you provide `xbar` as argument to an axis argument, see above.

The `xbar` style defines shifts if multiple plots are placed into one axis. If draws bars adjacent to each other, separated by `{`⟨*shift for multiple plots*⟩`}`. Furthermore, it sets the style `bar cycle list` and sets tick and legend appearance options.

The style is defined as follows.

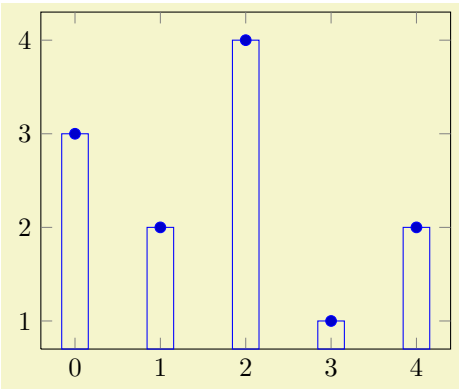
```
/pgfplots/xbar/.style={
    bar cycle list,
    tick align=outside,
    /pgfplots/legend image code/.code=
        {\draw[##1,bar width=3pt,yshift=-0.2em,bar shift=0pt]
            plot coordinates {(0cm,0.8em) (2*\pgfplotbarwidth,0.6em)};},
    /pgf/bar shift={%
            -0.5*(\numplots*\pgfplotbarwidth + (\numplots-1)*#1)  +
            (.5+\plotnum)*\pgfplotbarwidth + \plotnum*#1},
    /tikz/xbar},
```

The formular for `bar shift` assigns shifts dependend on the total number of plots and the current plot's number. It is designed to fill a total width of $n\cdot$`bar width`$+(n-1)\cdot${⟨*shift for multiple plots*⟩`}`. The 0.5 compensates for centering.

`/tikz/ybar` (no value)

`\addplot+[ybar]`

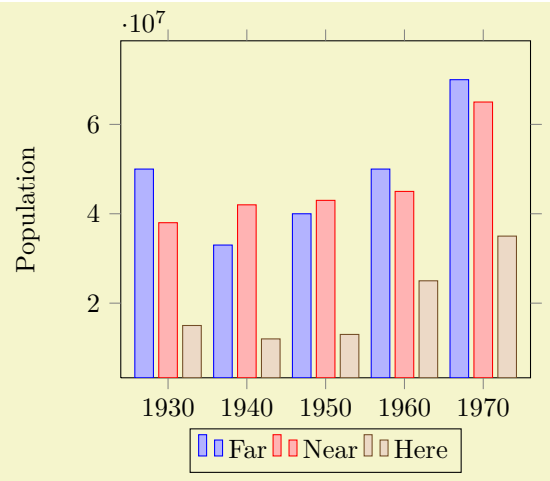
Like `xbar`, this option generates bar plots. It draws vertical bars between the ($x = 0$) line and each input coordinate.




```
\begin{tikzpicture}
\begin{axis}
\addplot+[ybar] plot coordinates
    {(0,3) (1,2) (2,4) (3,1) (4,2)};
\end{axis}
\end{tikzpicture}
```

The example above simply changes how input coordinates shall be visualized. As mentioned for `xbar`, one usually needs modified legends and shifts for multiple bars in the same axis.

There is a predefined style which installs these customizations when provided to the axis-environment:




```
\begin{tikzpicture}
\begin{axis}[
    x tick label style={
        /pgf/number format/1000 sep=},
    ylabel=Population,
    enlargelimits=0.15,
    legend style={at={(0.5,-0.15)},
        anchor=north,legend columns=-1},
    ybar,
    bar width=7pt,
]
\addplot
    coordinates {(1930,50e6) (1940,33e6)
         (1950,40e6) (1960,50e6) (1970,70e6)};

\addplot
    coordinates {(1930,38e6) (1940,42e6)
        (1950,43e6) (1960,45e6) (1970,65e6)};

\addplot
    coordinates {(1930,15e6) (1940,12e6)
        (1950,13e6) (1960,25e6) (1970,35e6)};
\legend{Far,Near,Here}
\end{axis}
\end{tikzpicture}
```

Here `ybar` yields `/pgfplots/ybar` because it is an argument to the axis, not to a single plot.

As for `xbar`, the bar width and shift can be configured with `bar width` and `bar shift`.

`/pgfplots/ybar={`⟨*shift for multiple plots*⟩`}` (style, default `2pt`)

As `/pgfplots/xbar`, this style sets the `/tikz/ybar` option to draw vertical bars, but it also provides commonly used options for vertical bars.

If you supply `ybar` to an axis environment, `/pgfplots/ybar` will be chosen instead of `/tikz/ybar`.

It changes the legend, draws ticks outside of the axis lines and draws multiple `\addplot` arguments adjacent to each other; block–centered at the x coordinate and separated by {⟨*shift for multiple plots*⟩}. Furthermore, it installs the style `bar cycle list`. It is defined similarly to `/pgfplots/xbar`.

`/tikz/bar width={`⟨*dimension*⟩`}` (no default, initially `10pt`)

Configures the width used by `xbar` and `ybar`. It is accepted to provide mathematical expressions.

`/tikz/bar shift={`⟨*dimension*⟩`}` (no default, initially `0pt`)

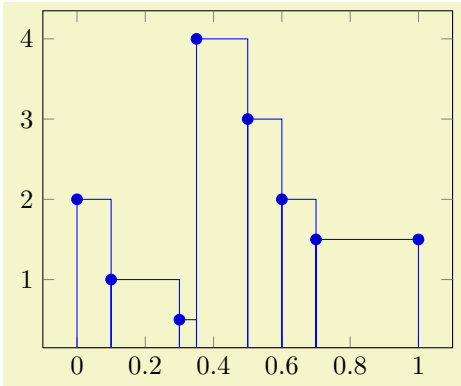
Configures a shift for `xbar` and `ybar`. Use `bar shift` together with `bar width` to draw multiple bar plots into the same axis. It is accepted to provide mathematical expressions.

`/tikz/ybar interval` (no value)

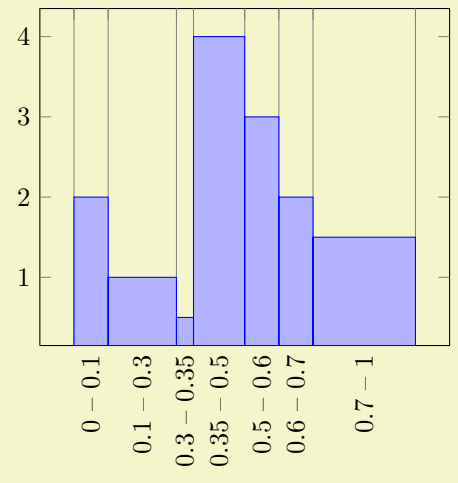
`\addplot+[ybar interval]`

This plot type produces vertical bars with width (and shift) relatively to intervals of coordinates.

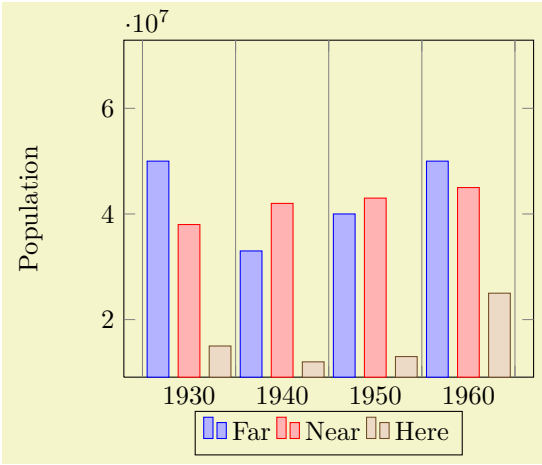
It is installed on a per-plot basis and configures *only* the visualization of coordinates. See the style `/pgfplots/ybar interval` which configures the appearance of the complete figure.




```
\begin{tikzpicture}
\begin{axis}
\addplot+[ybar interval] plot coordinates
    {(0,2) (0.1,1) (0.3,0.5) (0.35,4) (0.5,3)
     (0.6,2) (0.7,1.5) (1,1.5)};
\end{axis}
\end{tikzpicture}
```




```
\begin{tikzpicture}
\begin{axis}[ybar interval,
    xtick=data,
    xticklabel interval boundaries,
    x tick label style=
        {rotate=90,anchor=east}
    ]
\addplot coordinates
    {(0,2) (0.1,1) (0.3,0.5) (0.35,4) (0.5,3)
     (0.6,2) (0.7,1.5) (1,1.5)};
\end{axis}
\end{tikzpicture}
```




```
\begin{tikzpicture}
\begin{axis}[
    x tick label style={
        /pgf/number format/1000 sep=},
    ylabel=Population,
    enlargelimits=0.05,
    legend style={at={(0.5,-0.1)},
        anchor=north,legend columns=-1},
    ybar interval=0.7,
]
\addplot
    coordinates {(1930,50e6) (1940,33e6)
         (1950,40e6) (1960,50e6) (1970,70e6)};

\addplot
    coordinates {(1930,38e6) (1940,42e6)
        (1950,43e6) (1960,45e6) (1970,65e6)};

\addplot
    coordinates {(1930,15e6) (1940,12e6)
        (1950,13e6) (1960,25e6) (1970,35e6)};
\legend{Far,Near,Here}
\end{axis}
\end{tikzpicture}
```

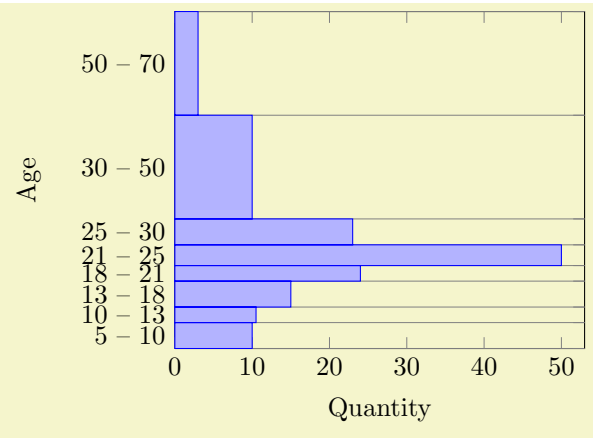
/pgfplots/ybar interval={⟨*relative width*⟩} (style, default 1)

A style which is intended to install options for `ybar interval` for a complete figure. This includes tick and legend appearance, management of multiple bar plots in one figure and a more adequate `cycle list` using the style `bar cycle list`.

/tikz/xbar interval (no value)

`\addplot+[xbar interval]`

As `ybar interval`, just for horizontal bars.




```
\begin{tikzpicture}
\begin{axis}[
    xmin=0,xmax=53,
    ylabel=Age,
    xlabel=Quantity,
    y label style={yshift=0.7cm},
    enlargelimits=false,
    ytick=data,
    yticklabel interval boundaries,
    xbar interval,
]
\addplot
    coordinates {(10,5) (10.5,10) (15,13)
        (24,18) (50,21) (23,25) (10,30)
        (3,50) (3,70)};
\end{axis}
\end{tikzpicture}
```

/pgfplots/xbar interval={⟨*relative width*⟩} (style, default 1)

A style which is intended to install options for `xbar interval` for a complete figure, see the style `/pgfplots/ybar interval` for details.

/pgfplots/xticklabel interval boundaries (no value)
/pgfplots/yticklabel interval boundaries (no value)

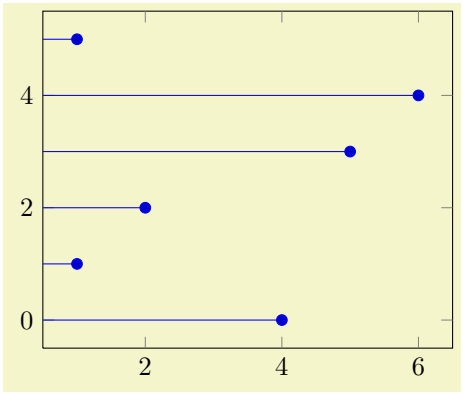
These are style keys which set `x tick label as interval` and configure the tick appearance to be {⟨*start*⟩} – {⟨*end*⟩} for each tick interval.

7.2.5 Comb Plots

Comb plots are very similar to bar plots except that they emplot single horizontal/vertical lines instead of rectangles.

/tikz/xcomb (no value)

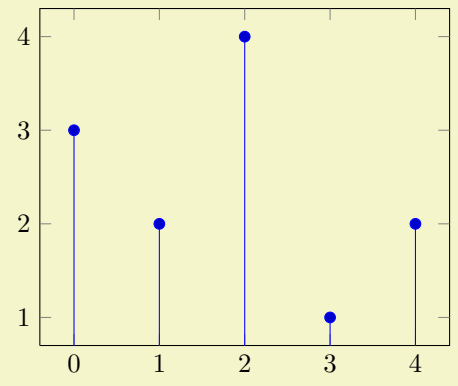
```
\addplot+[xcomb]
```




```
\begin{tikzpicture}
\begin{axis}
\addplot+[xcomb] coordinates
    {(4,0) (1,1) (2,2)
     (5,3) (6,4) (1,5)};
\end{axis}
\end{tikzpicture}
```

/tikz/ycomb (no value)

```
\addplot+[ycomb]
```

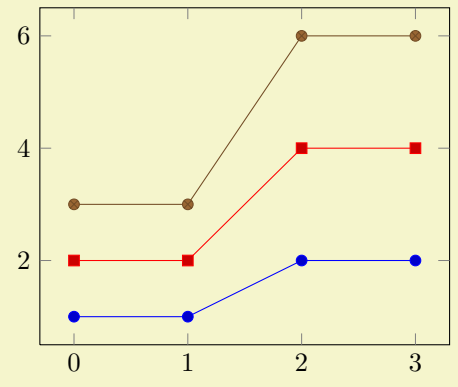



```
\begin{tikzpicture}
\begin{axis}
\addplot+[ycomb] plot coordinates
    {(0,3) (1,2) (2,4) (3,1) (4,2)};
\end{axis}
\end{tikzpicture}
```

7.2.6 Stacked Plots

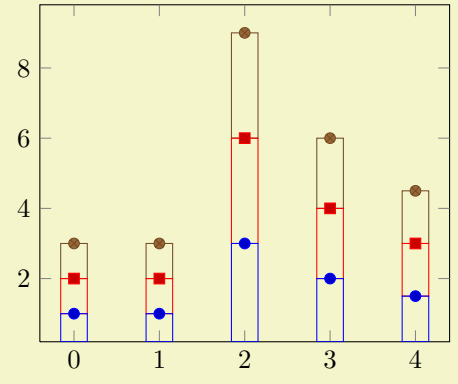
/pgfplots/stack plots=x|y|false (no default, initially false)

Allows stacking of plots in either x or y direction. Stacking means add either x- or y coordinates of successive \addplot commands on top of each other.

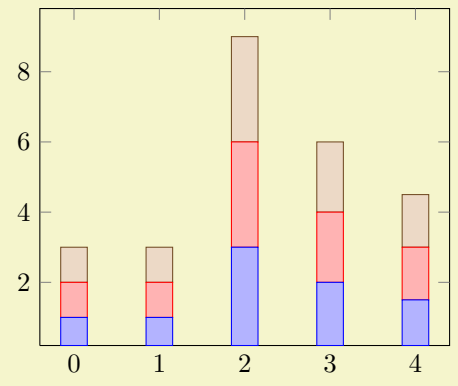



```
\begin{tikzpicture}
    \begin{axis}[stack plots=y]
    \addplot coordinates
        {(0,1) (1,1) (2,2) (3,2)};
    \addplot coordinates
        {(0,1) (1,1) (2,2) (3,2)};
    \addplot coordinates
        {(0,1) (1,1) (2,2) (3,2)};
    \end{axis}
\end{tikzpicture}
```

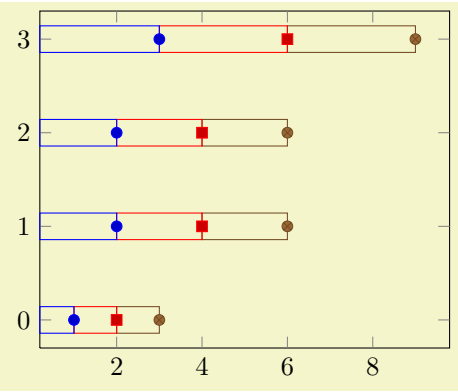
stack plots is particularly useful for bar plots. The following examples demonstrate its functionality. Normally, it is advisable to use the styles ybar stacked and xbar stacked which also set some other options.



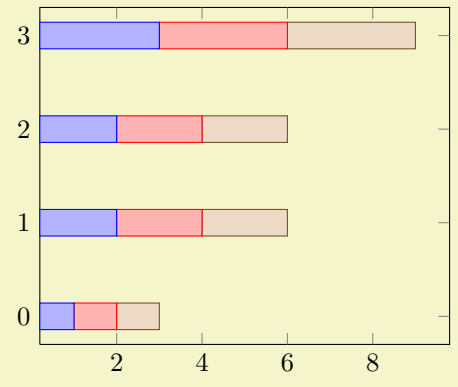

```
\begin{tikzpicture}
    \begin{axis}[stack plots=y,/tikz/ybar]
    \addplot coordinates
        {(0,1) (1,1) (2,3) (3,2) (4,1.5)};
    \addplot coordinates
        {(0,1) (1,1) (2,3) (3,2) (4,1.5)};
    \addplot coordinates
        {(0,1) (1,1) (2,3) (3,2) (4,1.5)};
    \end{axis}
\end{tikzpicture}
```




```
\begin{tikzpicture}
    \begin{axis}[ybar stacked]
    \addplot coordinates
        {(0,1) (1,1) (2,3) (3,2) (4,1.5)};
    \addplot coordinates
        {(0,1) (1,1) (2,3) (3,2) (4,1.5)};
    \addplot coordinates
        {(0,1) (1,1) (2,3) (3,2) (4,1.5)};
    \end{axis}
\end{tikzpicture}
```




```
\begin{tikzpicture}
    \begin{axis}[stack plots=x,/tikz/xbar]
    \addplot coordinates
        {(1,0) (2,1) (2,2) (3,3)};
    \addplot coordinates
        {(1,0) (2,1) (2,2) (3,3)};
    \addplot coordinates
        {(1,0) (2,1) (2,2) (3,3)};
    \end{axis}
\end{tikzpicture}
```




```
\begin{tikzpicture}
    \begin{axis}[xbar stacked]
    \addplot coordinates
        {(1,0) (2,1) (2,2) (3,3)};
    \addplot coordinates
        {(1,0) (2,1) (2,2) (3,3)};
    \addplot coordinates
        {(1,0) (2,1) (2,2) (3,3)};
    \end{axis}
\end{tikzpicture}
```

`/pgfplots/stack dir=plus|minus` (no default, initially `plus`)

Configures the direction of `stack plots`. The value `plus` will adds coordinates of successive plots while `minus` subtracts them.

`/pgfplots/reverse stacked plots=true|false` (no default, initially `true, default true`)

Configures the sequence in which stacked plots are drawn. This is more or less a technical detail which should not be changed in any normal case.

The motivation is as follows: suppose multiple `\addplot` commands are stacked on top of each other and they are processed in the order of appearance. Than, the second plot could easily draw its lines (or fill area) on top of the first one - hiding its marker or line completely. Therefor, PGFPLOTS reverses the sequence of drawing commands.

This has the side-effect that any normal Ti*k*Z-paths inside of an axis will also be processed in reverse sequence.

`/pgfplots/xbar stacked=plus|minus` (style, default `plus`)

A figure-wide style which enables stacked horizontal bars (i.e. `xbar` and `stack plots=x`). It also adjusts the legend and tick appearance and assigns a useful `cycle list`.

/pgfplots/ybar stacked=plus|minus (style, default plus)

A figure-wide style which enables stacked vertical bars (i.e. ybar and stack plots=y). It also adjusts the legend and tick appearance and assigns a useful cycle list.

/pgfplots/xbar interval stacked=plus|minus (style, default plus)

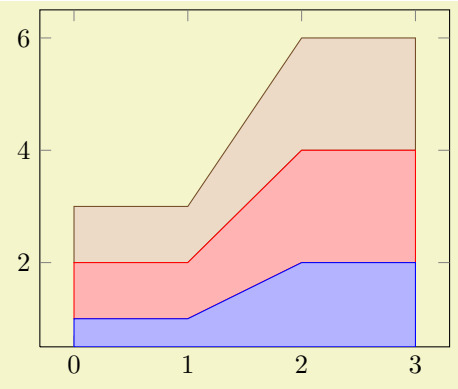
A sttyle similar to /pgfplots/xbar stacked for the interval based bar plot variant.

/pgfplots/ybar interval stacked=plus|minus (style, default plus)

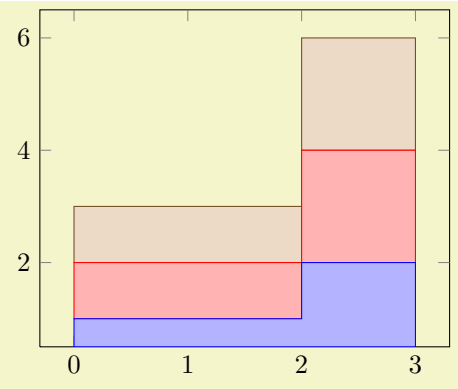
A sttyle similar to /pgfplots/ybar stacked for the interval based bar plot variant.

7.2.7 Area Plots

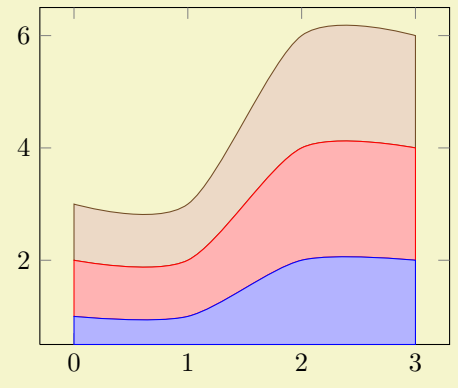
Area plots are a combination of \closedcycle and stack plots. They can be combined with any other plot type.




```
\begin{tikzpicture}
    \begin{axis}[
        stack plots=y,
        area cycle list]
    \addplot coordinates
        {(0,1) (1,1) (2,2) (3,2)}
        \closedcycle;
    \addplot coordinates
        {(0,1) (1,1) (2,2) (3,2)}
        \closedcycle;
    \addplot coordinates
        {(0,1) (1,1) (2,2) (3,2)}
        \closedcycle;
    \end{axis}
\end{tikzpicture}
```



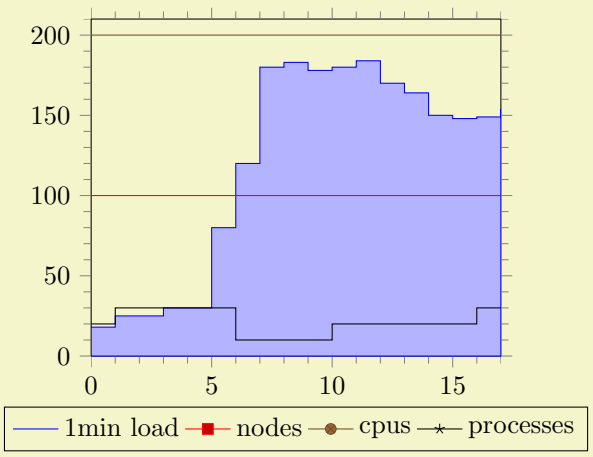

```
\begin{tikzpicture}
    \begin{axis}[
        const plot,
        stack plots=y,
        area cycle list]
    \addplot coordinates
        {(0,1) (1,1) (2,2) (3,2)}
        \closedcycle;
    \addplot coordinates
        {(0,1) (1,1) (2,2) (3,2)}
        \closedcycle;
    \addplot coordinates
        {(0,1) (1,1) (2,2) (3,2)}
        \closedcycle;
    \end{axis}
\end{tikzpicture}
```



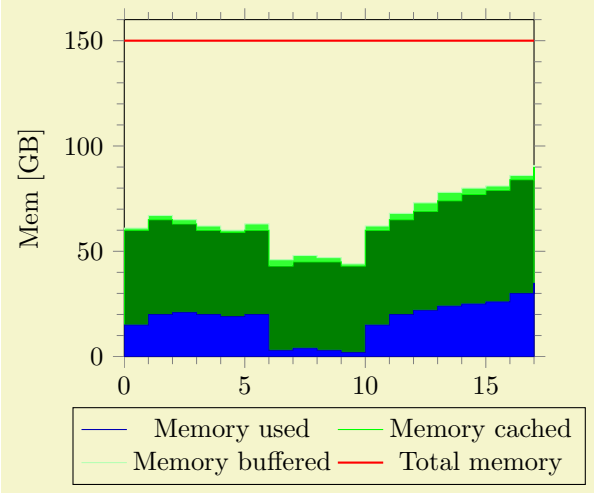

```
\begin{tikzpicture}
    \begin{axis}[
        smooth,
        stack plots=y,
        area cycle list]
    \addplot coordinates
        {(0,1) (1,1) (2,2) (3,2)}
        \closedcycle;
    \addplot coordinates
        {(0,1) (1,1) (2,2) (3,2)}
        \closedcycle;
    \addplot coordinates
        {(0,1) (1,1) (2,2) (3,2)}
        \closedcycle;
    \end{axis}
\end{tikzpicture}
```

time	1minload	nodes	cpus	processes	memused	memcached	membuf	memtotal
0	18	100	200	20	15	45	1	150
1	25	100	200	30	20	45	2	150
2	25	100	200	30	21	42	2	150
3	30	100	200	30	20	40	2	150
4	30	100	200	30	19	40	1	150
5	80	100	200	30	20	40	3	150
6	120	100	200	10	3	40	3	150
7	180	100	200	10	4	41	3	150
8	183	100	200	10	3	42	2	150
9	178	100	200	10	2	41	1	150
10	180	100	200	20	15	45	2	150
11	184	100	200	20	20	45	3	150
12	170	100	200	20	22	47	4	150
13	164	100	200	20	24	50	4	150
14	150	100	200	20	25	52	3	150
15	148	100	200	20	26	53	2	150
16	149	100	200	30	30	54	2	150
17	154	100	200	30	35	55	1	150

```
\pgfplotstableread{pgfplots.timeseries.dat}\table
\pgfplotstabletypeset\table
```




```
\pgfplotstableread
    {pgfplots.timeseries.dat}
    {\table}

\begin{tikzpicture}
    \begin{axis}[
        ymin=0,ymax=210,
        minor tick num=4,
        enlargelimits=false,
        tick align=outside,
        every axis plot post/.append style=
            {mark=none},
        const plot,
        legend style={
            at={(0.5,-0.15)},
            anchor=north,
            legend columns=-1}]

    \addplot[draw=blue,fill=blue!30!white]
     table[x=time,y=1minload] from \table
        \closedcycle;
    \addplot table[x=time,y=nodes] from \table;
    \addplot table[x=time,y=cpus] from \table;
    \addplot table[x=time,y=processes]
        from \table;
    \legend{1min load,nodes,cpus,processes}
    \end{axis}
\end{tikzpicture}
```



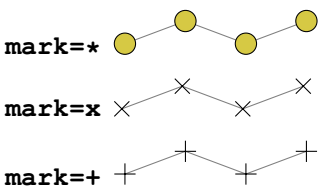

```
\pgfplotstableread{pgfplots.timeseries.dat}\table

\begin{tikzpicture}
    \begin{axis}[
        ymin=0,ymax=160,
        minor tick num=4,
        enlargelimits=false,
        tick align=outside,
        const plot,
        stack plots=y,
        cycle list={%
            {blue!70!black,fill=blue},%
            {green,fill=green!50!black},%
            {green!30!white,fill=green!80!white},%
            {red,thick}%
        },
        ylabel={Mem [GB]},
        legend style={
            at={(0.5,-0.15)},
            anchor=north,
            legend columns=2}]

    \addplot table[x=time,y=memused]      from \table \closedcycle;
    \addplot table[x=time,y=memcached]    from \table \closedcycle;
    \addplot table[x=time,y=membuf]       from \table \closedcycle;
    \addplot plot[/pgfplots/stack plots=false]
             table[x=time,y=memtotal]     from \table;
    \legend{Memory used,Memory cached,Memory buffered,Total memory}
    \end{axis}
\end{tikzpicture}
```

7.3 Markers and Linestyles

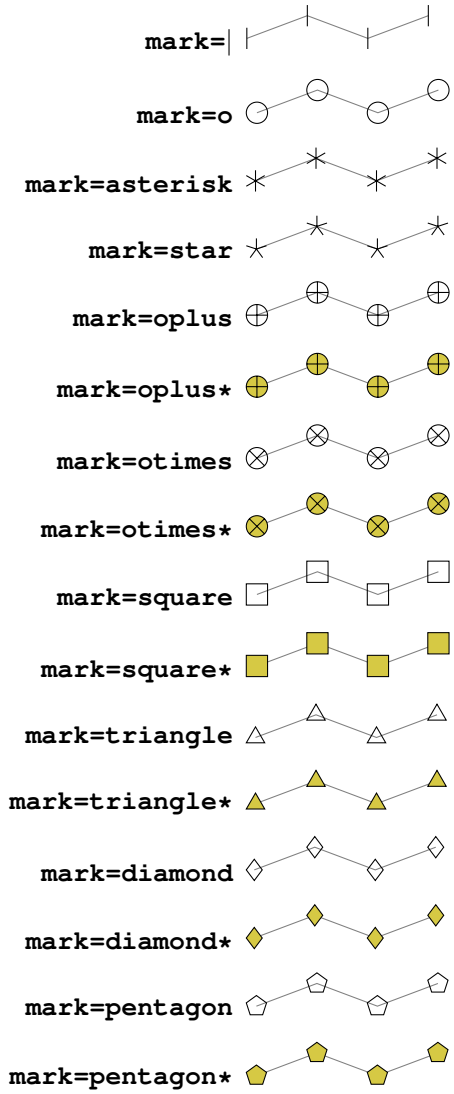
The following options of Ti*k*Z are available to plots.

7.3.1 Markers

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



And with `\usetikzlibrary{plotmarks}`:



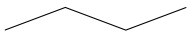
All these options have been drawn with the additional options

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

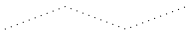
7.3.2 Line Styles

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

/tikz/solid (style, no value)



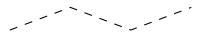
/tikz/dotted (style, no value)



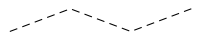
/tikz/densely dotted (style, no value)

/tikz/loosely dotted (style, no value)

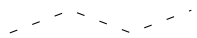
/tikz/dashed (style, no value)



/tikz/densely dashed (style, no value)



/tikz/loosely dashed (style, no value)



You may need the option `mark options={solid}` to avoid dotted or dashed marker boundaries.

7.3.3 Font Size and Line Width

Often, one wants to change line width and font sizes for plots. This can be done using the following options of Ti*k*Z.

/tikz/font={⟨*font name*⟩} (no default, initially `\normalfont`)

Sets the font which is to be used for text in nodes (like tick labels, legends or descriptions).

/tikz/line width={⟨*dimension*⟩} (no default, initially `0.4pt`)

Sets the line width. Please note that line widths for tick lines and grid lines are predefined, so you may need to override the styles `every tick` and `every axis grid`.

The `line width` key is changed quite often in Ti*k*Z. You should use

```
\pgfplotsset{every axis/.append style={line width=1pt}}
```

or

```
\pgfplotsset{every axis/.append style={thick}}
```

to change the overall line width. To also adjust ticks and grid lines, you can use

```
\pgfplotsset{every axis/.append style={
    line width=1pt,
    tick style={line width=0.6pt}}}
```

or styles like

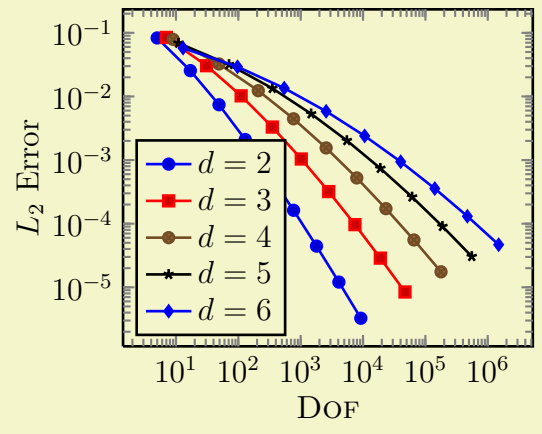
```
\pgfplotsset{every axis/.append style={
    thick,
    tick style={semithick}}}
```

The '`every axis plot`' style can be used to change line widths for plots only.

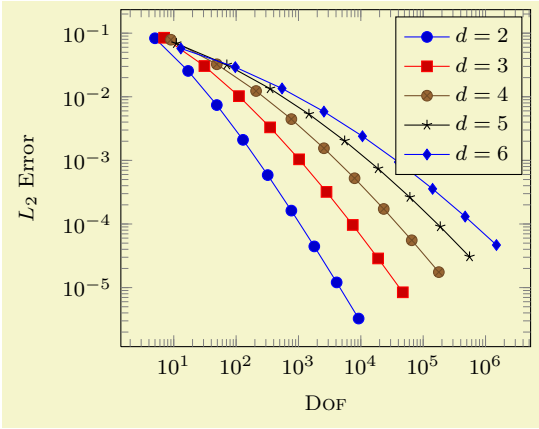
/tikz/ultra thin (no value)
/tikz/very thin (no value)
/tikz/semithick (no value)
/tikz/thick (no value)
/tikz/very thick (no value)
/tikz/ultra thick (no value)

These Ti*k*Z styles provide different predefined line widths.

This example shows the same plots as on page 8 (using `\plotcoords` as place holder for the commands on page 8), with different line width and font size.




```
\pgfplotsset{every axis/.append style={
    font=\large,
    line width=1pt,
    tick style={line width=0.8pt}}}
\begin{tikzpicture}
    \begin{loglogaxis}[
        legend style={at={(0.03,0.03)},
            anchor=south west},
        xlabel=\textsc{Dof},
        ylabel=$L_2$ Error
    ]
    \plotcoords
    \legend{$d=2$,$d=3$,$d=4$,$d=5$,$d=6$}
    \end{loglogaxis}
\end{tikzpicture}
```




```
\pgfplotsset{every axis/.append style={
    font=\footnotesize,
    thin,
    tick style={ultra thin}}}
\begin{tikzpicture}
    \begin{loglogaxis}[
        xlabel=\textsc{Dof},
        ylabel=$L_2$ Error
    ]
    \plotcoords
    \legend{$d=2$,$d=3$,$d=4$,$d=5$,$d=6$}
    \end{loglogaxis}
\end{tikzpicture}
```

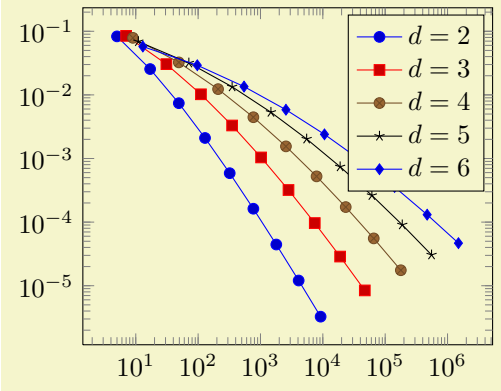
7.3.4 Options Controlling Linestyles

`/pgfplots/cycle list={`⟨*list*⟩`}` (no default)
`/pgfplots/cycle list name={`⟨*macro*⟩`}` (no default)

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

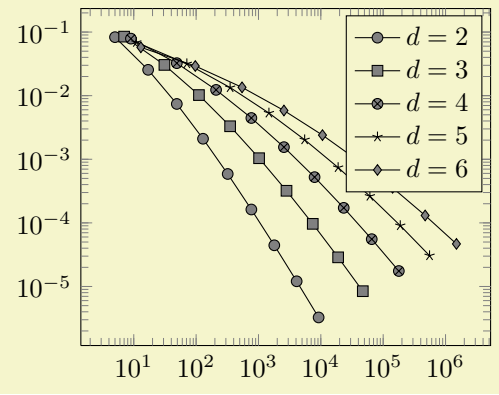
There are several possiblities to change it:

1. Use one of the predefined lists,



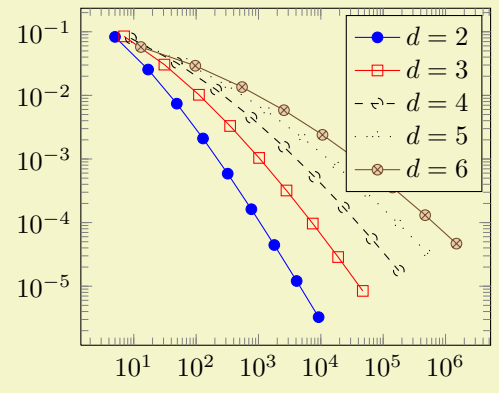

```
\begin{tikzpicture}
\begin{loglogaxis}[
    cycle list name=\coloredplotspeclist]
\plotcoords
\legend{$d=2$,$d=3$,$d=4$,$d=5$,$d=6$}
\end{loglogaxis}
\end{tikzpicture}
```

These examples employs the same coords as in the example on page 8.




```
\begin{tikzpicture}
\begin{loglogaxis}[
    cycle list name=\blackwhiteplotspeclist]
\plotcoords
\legend{$d=2$,$d=3$,$d=4$,$d=5$,$d=6$}
\end{loglogaxis}
\end{tikzpicture}
```

2. Provide the list explicitly,




```
\begin{tikzpicture}
\begin{loglogaxis}[cycle list={%
    {blue,mark=*},
    {red,mark=square},
    {dashed,mark=o},
    {loosely dotted,mark=+},
    {brown!60!black,
        mark options={fill=brown!40},
        mark=otimes*}}
]
\plotcoords
\legend{$d=2$,$d=3$,$d=4$,$d=5$,$d=6$}
\end{loglogaxis}
\end{tikzpicture}
```

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

3. Define macro names and use them with '`cycle list name`':

```
\pgfcreateplotcyclelist{\mylist}{%
    {blue,mark=*},
    {red,mark=square},
    {dashed,mark=o},
    {loosely dotted,mark=+},
    {brown!60!black,mark options={fill=brown!40},mark=otimes*}}
}
...
\begin{axis}[cycle list name=\mylist]
    ...
\end{axis}
```

Remark: You can also terminate single entries with '`\\`' as in

```
\begin{axis}[cycle list={%
    blue,mark=*\\%
    red,mark=square\\%
    dashed,mark=o\\%
    loosely dotted,mark=+\\%
    brown!60!black,
        mark options={fill=brown!40},
        mark=otimes*\\}
]
...
\end{axis}
```

In this case, the *last* entry also needs a terminating '`\\`', but you can omit braces around the single entries.

7.4 Axis Descriptions

7.4.1 Labels

/pgfplots/xlabel={⟨*text*⟩} (no default)
/pgfplots/ylabel={⟨*text*⟩} (no default)

The options `xlabel` and `ylabel` change axis labels to {⟨*text*⟩} which is any TEX text. Use `xlabel={, = characters}` if you need to include '=' or ',' literally.

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

The default styles are

```
\pgfplotsset{every axis label/.style={}}
\pgfplotsset{every axis x label/.style={
    at={(0.5,0)},
    below,
    yshift=-15pt}}
\pgfplotsset{every axis y label/.style={
    at={(0,0.5)},
    xshift=-35pt,
    rotate=90}}
```

Please add options using `.append style` instead of overwriting the default styles to ensure compatibility with future versions.

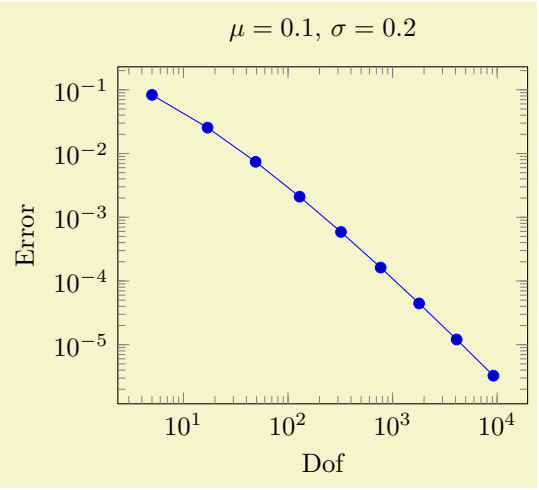
```
\pgfplotsset{every axis label/.append style={...}}
\pgfplotsset{every axis x label/.append style={...}}
\pgfplotsset{every axis y label/.append style={...}}
```

/pgfplots/title={⟨*text*⟩} (no default)

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.




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

    \addplot 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}%
```

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

```
\pgfplotsset{every axis title/.append style={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.

7.4.2 Legend

`/pgfplots/legend columns={`⟨*number*⟩`}` (default `1`)

Allows to configure the maximum number of adjacent legend entries. The default value `1` places legend entries vertically below each other.

Use `legend columns=-1` to draw all entries horizontally.

`/pgfplots/legend plot pos=left|right|none` (no default, initially `left`)

Configures where the small line specifications will be drawn: left of the description, right of the description or not at all.

`/pgfplots/legend image code/.code={`⟨`...`⟩`}`

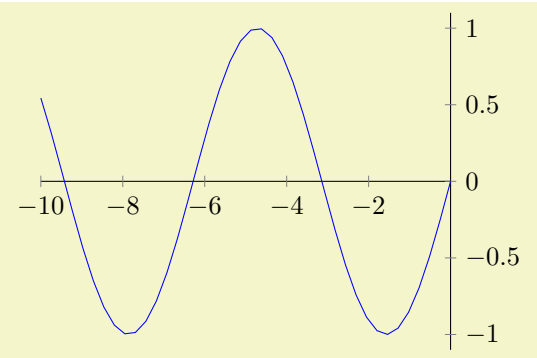
Allows to replace the default images which are drawn inside of legends. The first argument to this option is the plot specification, a key-value list which has been determined by `\addplot`.

The default is

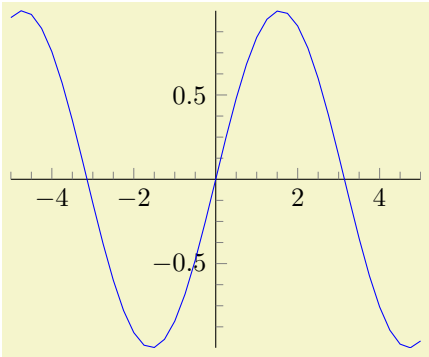
```
/pgfplots/legend image code/.code={%
    \draw[#1,mark repeat=2,mark phase=2]
        plot coordinates {
            (0cm,0cm)
            (0.3cm,0cm)
            (0.6cm,0cm)%
        };%
}
```

7.4.3 Axis Lines

By default the axis lines are drawn as a `box`, but it is possible to change the appearance of the x and y axis lines.

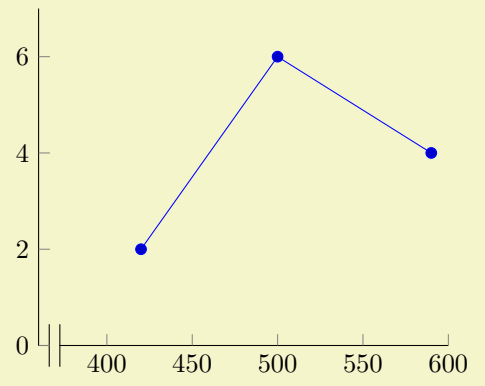



```
\begin{tikzpicture}
\begin{axis}[
    axis x line=middle,
    axis y line=right,
    tick align=center,
    ymax=1.1, ymin=-1.1,
    enlargelimits=false
]
    \addplot[blue,mark=none]
        plot[id=sinneg,domain=-10:0,samples=40]
        function{sin(x)};
\end{axis}
\end{tikzpicture}
```

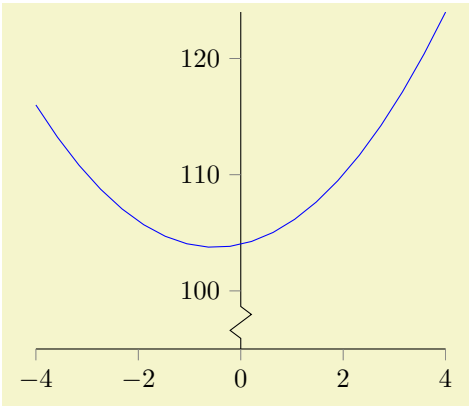



```
\begin{tikzpicture}
\begin{axis}[
    enlargelimits=false,
    minor tick num=3,
    axis y line=center,
    axis x line=middle
    ]
    \addplot[blue,mark=none]
        plot[domain=-5:5,samples=40]
        (\x,{sin(\x r)});
\end{axis}
\end{tikzpicture}
```

In case the range of either of the axis do not include the zero value, it is possible to visualize this with a discontinuity decoration on the corresponding axis line.




```
\begin{tikzpicture}
\begin{axis}[
    axis x line=bottom,
    axis x discontinuity=parallel,
    axis y line=left,
    xmin=360, xmax=600,
    ymin=0, ymax=7,
     enlargelimits=false
]
    \addplot coordinates {
        (420,2)
        (500,6)
        (590,4)
    };
\end{axis}
\end{tikzpicture}
```




```
\begin{tikzpicture}
\begin{axis}[
    axis x line=bottom,
    axis y line=center,
    tick align=outside,
    axis y discontinuity=crunch,
    ymin=95, enlargelimits=false
]
    \addplot[blue,mark=none]
        plot[id=square,domain=-4:4,samples=20]
        function{x*x+x+104};
\end{axis}
\end{tikzpicture}
```

/pgfplots/axis x line=box|top|middle|bottom|none (no default, initially box)

Allows to choose the location of the x axis line(s). The labels and ticks are placed accordingly. Bottom will draw only a line at y=ymin, middle will draw a line at y=0, and top will draw only a line at y=ymax. Box is a combination of options top and bottom.

/pgfplots/axis y line=box|left|center|right|none (no default, initially box)

Allows to choose the location of the y axis line(s). The labels and ticks are placed accordingly. Left will draw only a line at x=xmin, center will draw a line at x=0, and right will draw only a line at x=xmax. Box is a combination of options left and right.

/pgfplots/axis x discontinuity=crunch|parallel|none (no default, initially none)

Insert a discontinuity decoration on the x axis. This is to visualize that the y axis does cross the x axis at its 0 value, because the minimum x axis value is positive or the maximum value is negative.

/pgfplots/axis y discontinuity=crunch|parallel|none (no default, initially none)

Similar to axis x discontinuity, but now for the y axis.

/pgfplots/hide axis=true|false (no default, initially 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.

7.5 Scaling Options

/pgfplots/width={⟨*dimen*⟩} (no default)

Sets the width of the final picture to {⟨*dimen*⟩}. If no height is specified, scaling will respect aspect ratios.

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, but their size is used to determine the axis scaling.

- You can use the `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}
```

- 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[6]. This may be fixed in future versions.
  Use the `x={⟨dimension⟩}`, `y={⟨dimension⟩}` and `scale only axis` options if the scaling happens to be wrong.

`/pgfplots/height={⟨dimen⟩}` (no default)

See `width`.

`/pgfplots/scale only axis=true|false` (no default, initially `false`)

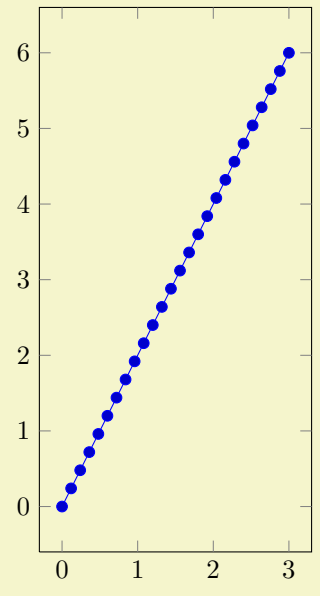
If `scale only axis` is enabled, label, tick and legend dimensions won't influence the size of the axis rectangle, that means `width` and `height` apply only to the axis rectangle

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

`/pgfplots/x={⟨dimen⟩}` (no default)
`/pgfplots/y={⟨dimen⟩}` (no default)

Sets the unit size for one x (or y) coordinate to `{⟨dimen⟩}`.

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




```
\begin{tikzpicture}
\begin{axis}[x=1cm,y=1cm]
\addplot plot[domain=0:3] (\x,2*\x);
\end{axis}
\end{tikzpicture}
```

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

Setting `x` and/or `y` for logarithmic axis will set the dimension used for $1 \cdot e \approx 2.71828$.

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 rescales its coordinates (see the `width` option).

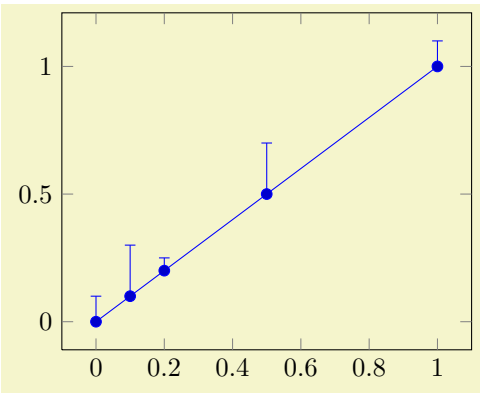
7.6 Error Bars

PGFPLOTS supports error bars for normal and logarithmic plots.

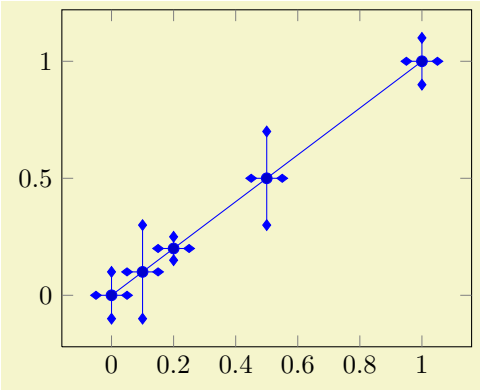
Error bars are enabled for each plot separately, using ⟨*behavior options*⟩ after `\addplot`:

```
\addplot plot[/pgfplots/error bars/.cd,x dir=both,y dir=both] ...
```

Error bars inherit all drawing options of the associated plot, but they use their own marker and style arguments additionally.

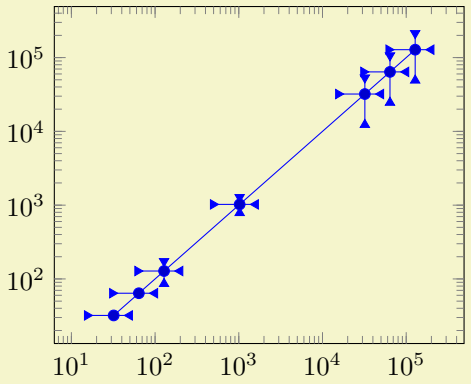



```
\begin{tikzpicture}
\begin{axis}
\addplot plot[/pgfplots/error bars/.cd,
    y dir=plus,y explicit]
coordinates {
    (0,0)     +- (0.5,0.1)
    (0.1,0.1) +- (0.05,0.2)
    (0.2,0.2) +- (0,0.05)
    (0.5,0.5) +- (0.1,0.2)
    (1,1)     +- (0.3,0.1)};
\end{axis}
\end{tikzpicture}
```

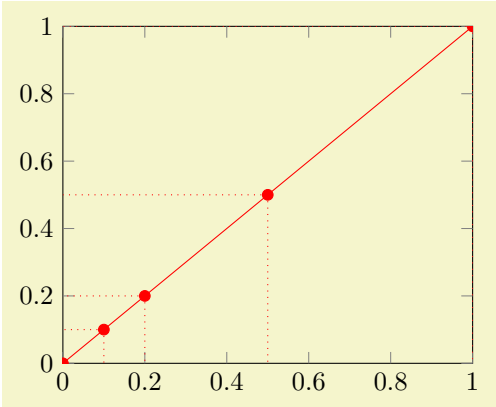



```
\begin{tikzpicture}
\begin{axis}
\addplot plot[/pgfplots/error bars/.cd,
    y dir=both,y explicit,
    x dir=both,x fixed=0.05,
    error mark=diamond*]
coordinates {
    (0,0)     +- (0.5,0.1)
    (0.1,0.1) +- (0.05,0.2)
    (0.2,0.2) +- (0,0.05)
    (0.5,0.5) +- (0.1,0.2)
    (1,1)     +- (0.3,0.1)};
\end{axis}
\end{tikzpicture}
```

x	y	errorx	errory
32	32	0	0
64	64	0	0
128	128	0	0.3
1,024	1,024	0	0.2
32,068	32,068	0	0.6
64,000	64,000	0	0.6
$1.28 \cdot 10^5$	$1.28 \cdot 10^5$	0	0.6




```
\pgfplotstabletypesetfile{pgfplots.testtable2.dat}

\begin{tikzpicture}
\begin{loglogaxis}
\addplot plot[/pgfplots/error bars/.cd,
    x dir=both,x fixed relative=0.5,
    y dir=both,y explicit relative,
    error mark=triangle*]
    table[x=x,y=y,y error=errory]
    {pgfplots.testtable2.dat};
\end{loglogaxis}
\end{tikzpicture}
```




```
\begin{tikzpicture}
\begin{axis}[enlargelimits=false]
\addplot[red,mark=*]
    plot[/pgfplots/error bars/.cd,
    y dir=minus,y fixed relative=1,
    x dir=minus,x fixed relative=1,
    error mark=none,
    error bar style={dotted}]
coordinates
    {(0,0) (0.1,0.1) (0.2,0.2)
     (0.5,0.5) (1,1)};
\end{axis}
\end{tikzpicture}
```

`/pgfplots/error bars/x dir`=none|plus|minus|both (no default, initially none)
`/pgfplots/error bars/y dir`=none|plus|minus|both (no default, initially none)

Draws either no error bars at all, only marks at $x + \epsilon_x$, only marks at $x - \epsilon_x$ or marks at both, $x + \epsilon_x$ and $x - \epsilon_x$. The x-error ϵ_x is acquired using one of the following options.

The same holds for the `y dir` option.

`/pgfplots/error bars/x fixed`={⟨*value*⟩} (no default, initially 0)
`/pgfplots/error bars/y fixed`={⟨*value*⟩} (no default, initially 0)

Provides a common, absolute error $\epsilon_x = \langle value \rangle$ for all input coordinates.

For linear x axes, the error mark is drawn at $x \pm \epsilon_x$ while for logarithmic x axes, it is drawn at $\log(x \pm \epsilon_x)$. Computations are performed in PGF's floating point arithmetics.

`/pgfplots/error bars/x fixed relative`={⟨*percent*⟩} (no default, initially 0)
`/pgfplots/error bars/y fixed relative`={⟨*percent*⟩} (no default, initially 0)

Provides a common, relative error $\epsilon_x = \langle percent \rangle \cdot x$ for all input coordinates. The argument ⟨*percent*⟩ is thus given relatively to input x coordinates such that $\langle percent \rangle = 1$ means 100%.

Error marks are thus placed at $x \cdot (1 \pm \epsilon_x)$ for linear axes and at $\log(x \cdot (1 \pm \epsilon_x))$ for logarithmic axes. Computations are performed in floating point for linear axis and using the identity $\log(x \cdot (1 \pm \epsilon_x)) = \log(x) + \log(1 \pm \epsilon_x)$ for logarithmic scales.

`/pgfplots/error bars/x explicit` (no value)
`/pgfplots/error bars/y explicit` (no value)

Configures the error bar algorithm to draw x-error bars at any input coordinate for which user-specified errors are available. Each error is interpreted as absolute error, see `x fixed` for details.

The different input formats of errors are described in section 7.6.1.

`/pgfplots/error bars/x explicit relative` (no value)
`/pgfplots/error bars/y explicit relative` (no value)

Configures the error bar algorithm to draw x-error bars at any input coordinate for which user-specified errors are available. Each error is interpreted as relative error, that means error marks are placed at $x(1 \pm \langle value \rangle(x))$ (works as for `error bars/x fixed relative`).

`/pgfplots/error bars/error mark`=⟨*marker*⟩ (no default)

Sets an error marker for any error bar. {⟨*marker*⟩} is expected to be a valid plot mark, see section 7.3.

`/pgfplots/error bars/error mark options`={⟨*key-value-list*⟩} (no default)

Sets a key-value list of options for any error mark. This option works similary to the Ti*k*Z 'mark options' key.

`/pgfplots/error bars/error bar style`={⟨*key-value-list*⟩} (no default)

Appends the argument to '/pgfplots/every error bar' which is installed at the beginning of every error bar.

`/pgfplots/error bars/draw error bar/.code 2 args`={⟨...⟩}

Allows to change the default drawing commands for error bars. The two arguments are

- the source point, (x, y) and
- the target point, $(\tilde{x}, \tilde{y})$.

Both are determined by PGFPLOTS according to the options described above. The default code is

```
[basicstyle=\footnotesize\ttfamily]
/pgfplots/error bars/draw error bar/.code 2 args={%
    \pgfkeysgetvalue{/pgfplots/error bars/error mark}%
        {\pgfplotserrorbarsmark}%
    \pgfkeysgetvalue{/pgfplots/error bars/error mark options}%
        {\pgfplotserrorbarsmarkopts}%
    \draw #1 -- #2 node[pos=1,sloped,allow upside down] {%
        \expandafter\tikz\expandafter[\pgfplotserrorbarsmarkopts]{%
            \expandafter\pgfuseplotmark\expandafter{\pgfplotserrorbarsmark}%
            \pgfusepath{stroke}}%
    };
}
```

7.6.1 Input Formats of Error Coordinates

Error bars with explicit error estimations for single data points require some sort of input format. This applies to 'error bars/⟨*[xy]*⟩ explizit' and 'error bars/⟨*[xy]*⟩ explizit relative'.

Error bar coordinates can be read from 'plot coordinates' or from 'plot table'. The inline plot coordinates format is

```
\addplot coordinates {
    (1,2) +- (0.4,0.2)
    (2,4) +- (1,0)
    (3,5)
    (4,6) +- (0.3,0.001)
}
```

where $(1, 2) \pm (0.4, 0.2)$ is the first coordinate, $(2, 4) \pm (1, 0)$ the second and so forth. The point $(3, 5)$ has no error coordinate.

The ‘plot table’ format is

```
\addplot table[x error=COLNAME,y error=COLNAME]
```

or

```
\addplot table[x error index=COLINDEX,y error index=COLINDEX]
```

These options are used as the ‘x’ and ‘x index’ options.

You can supply error coordinates even if they are not used at all; they will be ignored silently in this case.

7.7 Number Formatting Options

PGFPLOTS typeset tick labels rounded to given precision and in configurable number formats. The command to do so is \pgfmathprintnumber; it uses the current set of number formatting options.

These options are described in all detail in the manual for PGFPLOTSTABLE, which comes with PGFPLOTS. Please refer to that manual.

\pgfmathprintnumber{⟨x⟩}

Generates pretty-printed output for the (real) number {⟨x⟩}. The input number {⟨x⟩} is parsed using \pgfmathfloatparsenumber which allows arbitrary precision.

Numbers are typeset in math mode using the current set of number printing options, see below. Optional arguments can also be provided using \pgfmathprintnumber[⟨options⟩]{⟨x⟩}.

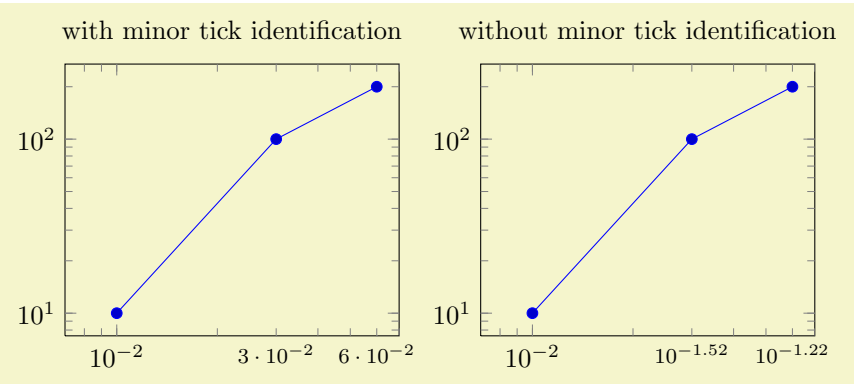
Please refer to the manual of PGFPLOTSTABLE (shipped with this package) for details about the number options.

/pgfplots/log identify minor tick positions=true|false (no default, initially true)

Set this to true if you want to identify log–plot tick labels at positions

$$i \cdot 10^j$$

with $i \in \{2, 3, 4, 5, 6, 7, 8, 9\}$, $j \in \mathbb{Z}$. This may be valueable in conjunction with the ‘extra x ticks’ and ‘extra y ticks’ options. An example is shown below: The axis range is so small that only one tick label 10^j is inside of it. Extra tick labels can be placed on top of the normal ticks, and placing ticks at $i \cdot 10^j$ may be appropriate.



```
\pgfplotsset{every axis/.append style={%
    width=6cm,
    xmin=7e-3,xmax=7e-2,
    extra x ticks={3e-2,6e-2},
    extra x tick style={major tick length=0pt,font=\footnotesize}
}}%
\begin{tikzpicture}%
    \begin{loglogaxis}[
        title=with minor tick identification]
    \addplot coordinates {
        (1e-2,10)
        (3e-2,100)
        (6e-2,200)
    };
    \end{loglogaxis}
\end{tikzpicture}%

\begin{tikzpicture}%
    \begin{loglogaxis}[
        title=without minor tick identification,
        log identify minor tick positions=false]
    \addplot coordinates {
        (1e-2,10)
        (3e-2,100)
        (6e-2,200)
    };
    \end{loglogaxis}%
\end{tikzpicture}%
```

/pgfplots/log number format code/.code={⟨...⟩}

Provides TEX-code to generate log plot tick labels. Argument '#1' is the (natural) logarithm of the tick position. The default implementation invokes `log base 10 number format code` after it changed the log basis to 10. It also checks the other log plot options.

/pgfplots/log base 10 number format code/.code={⟨...⟩}

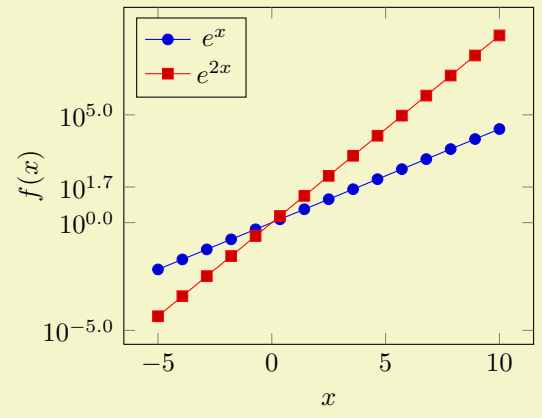
Allows to change the overall appearance of base 10 log plot tick labels. The default is

```
log base 10 number format code/.code={%
    $10^{\pgfmathprintnumber{#1}}$}
```

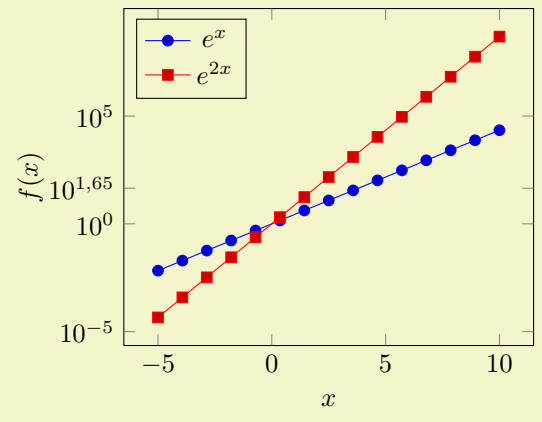
where the '`log plot exponent style`' allows to change number formatting options.

/pgfplots/log plot exponent style={⟨*key-value-list*⟩} (no default)

Allows to configure the number format of log plot exponents. This style is installed just before '`log base 10 number format code`' will be invoked. Please note that this style will be installed within the default code for '`log number format code`'.




```
\tikzset{samples=15}
\pgfplotsset{every axis/.append style={
        width=7cm,
        xlabel=$x$,
        ylabel=$f(x)$,
        extra y ticks={45},
        legend style={at={(0.03,0.97)},
            anchor=north west}}}

\begin{tikzpicture}
\begin{semilogyaxis}[
    log plot exponent style/.style={
        /pgf/number format/fixed zerofill,
        /pgf/number format/precision=1}]

    \addplot plot[id=gnuplot_exp,domain=-5:10]
        function{exp(x)};
    \addplot plot[id=gnuplot_expv,domain=-5:10]
        function{exp(2*x)};

    \legend{$e^x$,$e^{2x}$}
\end{semilogyaxis}
\end{tikzpicture}
```




```
\tikzset{samples=15}
\pgfplotsset{every axis/.append style={
        width=7cm,
        xlabel=$x$,
        ylabel=$f(x)$,
        extra y ticks={45},
        legend style={at={(0.03,0.97)},
            anchor=north west}}}

\begin{tikzpicture}
\begin{semilogyaxis}[
    log plot exponent style/.style={
        /pgf/number format/fixed,
        /pgf/number format/use comma,
        /pgf/number format/precision=2}]

    \addplot plot[id=gnuplot_exp,domain=-5:10]
        function{exp(x)};
    \addplot plot[id=gnuplot_expv,domain=-5:10]
        function{exp(2*x)};

    \legend{$e^x$,$e^{2x}$}
\end{semilogyaxis}
\end{tikzpicture}
```

7.8 Specifying the Plotted Range

/pgfplots/xmin={⟨*coord*⟩} (no default)
/pgfplots/ymin={⟨*coord*⟩} (no default)
/pgfplots/xmax={⟨*coord*⟩} (no default)
/pgfplots/ymax={⟨*coord*⟩} (no default)

The options `xmin`, `xmax` and `ymin`, `ymax` allow to define the axis limits, i.e. the lower left and the upper right corner. Everything outside of the axis limits will be clipped away.

If one of `xmin` or `xmax` is missing, the x-interval will be determined automatically. The same holds true if one of `ymin` or `ymax` is missing: in this case, the y-interval will be determined automatically.

If x-limits have been specified explicitly and y-limits are computed automatically, the automatic computation of y-limits will only considers points which fall into the specified x-range (and vice–versa). This can be disable using `clip limits=false`.

Axis limits can be increased automatically using the `enlargelimits` option.

/pgfplots/xmode=normal|linear|log (no default, initially `normal`)
/pgfplots/ymode=normal|linear|log (no default, initially `normal`)

Allows to choose between linear (=normal) or logarithmic axis scaling or logplots for each x, y-combination.

/pgfplots/clip limits=true|false (no default, initially `true`)

Configures what to do if some, but not all axis limits have been specified explicitly. In case `clip limits=true`, the automatic limit computation will *only* consider points which do not contradict the explicitly set limits.

This option has nothing to do with path clipping, it only affects how the axis limits are computed.

/pgfplots/enlargelimits=true|false|auto|{⟨*val*⟩} (no default)

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).

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

7.9 Tick and Grid Options

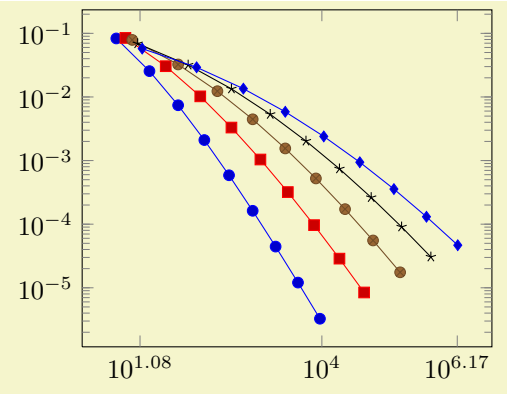
`/pgfplots/xtick=\empty|data|{`⟨*coordinate list*⟩`}` (no default, initially {⟨⟩})
`/pgfplots/ytick=\empty|data|{`⟨*coordinate list*⟩`}` (no default, initially {⟨⟩})

The options `xtick` and `ytick` assigns a list of *Positions* where ticks shall be placed. The argument is either the command `\empty`, `data` or a list of coordinates. The choice `\empty` will result in no tick at all. The special value `data` will produce tick marks at every coordinate of the first plot. Otherwise, tick marks will be placed at every coordinate in {⟨*coordinate list*⟩}. If this list is empty, PGFPLOTS will compute a default list.

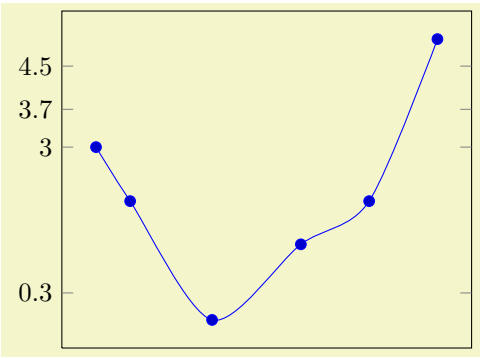
{⟨*coordinate list*⟩} will be used inside of a `\foreach \x in {`⟨*coordinate list*⟩`}` statement. The format is as follows:

- `{0,1,2,5,8,1e1,1.5e1}` (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(·) to each element in '{⟨*coordinate list*⟩}'.




```
\begin{tikzpicture}
    \begin{loglogaxis}[xtick={12,9897,1468864}]
    \plotcoords
    \end{loglogaxis}
\end{tikzpicture}
```




```
\begin{tikzpicture}
\begin{axis}[
    xtick=\empty,
    ytick={-2,0.3,3,3.7,4.5}]
\addplot+[smooth] coordinates {
    (-2,3) (-1.5,2) (-0.3,-0.2)
    (1,1.2) (2,2) (3,5)};
\end{axis}
\end{tikzpicture}
```

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 is to place ticks at positions 10^i in the axis' range. Which positions depends on the axis scaling and the dimensions of the picture. The default tick positions can be reconfigured with

- '`max space between ticks={`⟨*number*⟩`}`' 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. The default is 35.
- '`try min ticks={`⟨*number*⟩`}`' configures a loose lower bound on the number of ticks. It should be considered as a suggestion, not a tight limit. The default is 4. This number will increase the number of ticks if '`max space between ticks`' produces too few of them.
- '`try min ticks log={`⟨*number*⟩`}`' The same for logarithmic axis.

The total number of ticks may still vary because not all fractional numbers in the axis' range are valid tick positions.

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

```
\pgfplotsset{every tick/.style={very thin,gray}}
\pgfplotsset{every minor tick/.style={}}
```

or

```
\pgfplotsset{every tick/.append style={very thin,gray}}
\pgfplotsset{every minor tick/.append style={black}}
```

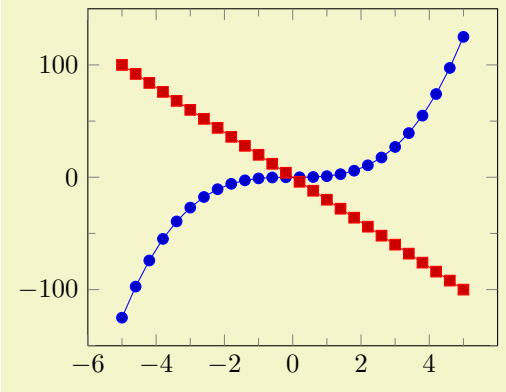
Please prefer the '`.append style`' versions to ensure compatibility with future versions.

This style commands can be used at any time. The tick line width can be configured with '`major tick length`' and '`minor tick length`'.

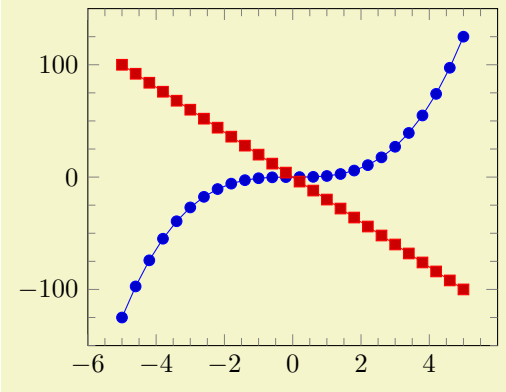
`/pgfplots/minor tick num={`⟨*number*⟩`}` (no default)

Sets both, `minor x tick num` and `minor y tick num` to {⟨*number*⟩}.

Minor ticks will be disabled if the major ticks don't have the same distance.




```
\begin{tikzpicture}
    \begin{axis}[minor tick num=1]
    \addplot (\x,\x^3);
    \addplot (\x,-20*\x);
    \end{axis}
\end{tikzpicture}
```



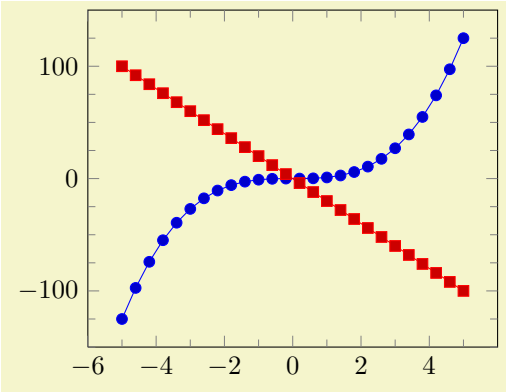

```
\begin{tikzpicture}
    \begin{axis}[minor tick num=3]
    \addplot (\x,\x^3);
    \addplot (\x,-20*\x);
    \end{axis}
\end{tikzpicture}
```

`/pgfplots/minor x tick num={`⟨*number*⟩`}` (no default, initially `0`)

/pgfplots/minor y tick num={⟨*number*⟩} (no default, initially 0)

Sets the number of minor tick lines used for linear x or y axis separately.

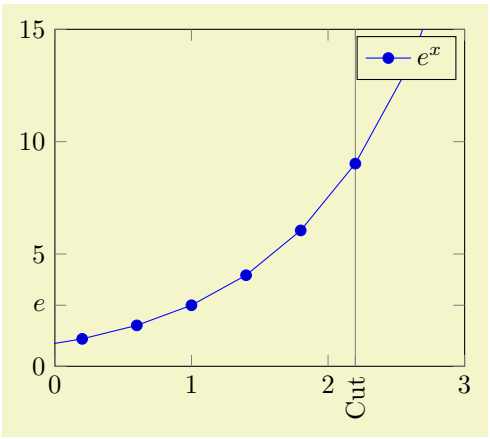
Minor ticks will be disabled if the major ticks don't have the same distance.



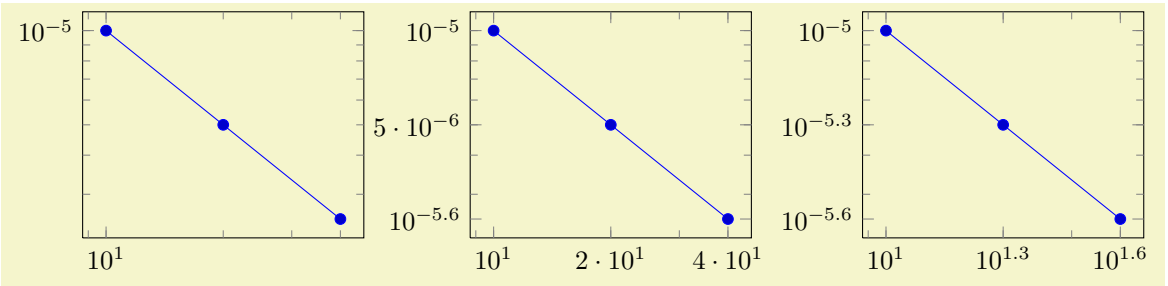

```
\begin{tikzpicture}
    \begin{axis}[minor x tick num=1,
                 minor y tick num=3]
    \addplot (\x,\x^3);
    \addplot (\x,-20*\x);
    \end{axis}
\end{tikzpicture}
```

/pgfplots/extra x ticks={⟨*coordinate list*⟩} (no default)
/pgfplots/extra y ticks={⟨*coordinate list*⟩} (no default)

Adds *additional* tick positions and tick labels to the x or y axis. 'Additional' tick positions do not affect the normal tick placement algorithms, they are drawn after the normal ticks. This has two benefits: first, you can add single, important tick positions without disabling the default tick label generation and second, you can draw tick labels 'on top' of others, possibly using different style flags.




```
\begin{tikzpicture}
\begin{axis}[
    xmin=0,xmax=3,ymin=0,ymax=15,
    extra y ticks={2.71828},
    extra y tick labels={$e$},
    extra x ticks={2.2},
    extra x tick style={grid=major,
        /pgfplots/tick label style={
            rotate=90,anchor=east}},
    extra x tick labels={Cut},
]
    \addplot (\x,{exp(\x)});
    \addlegendentry{$e^x$}
\end{axis}
\end{tikzpicture}
```



```
\pgfplotsset{every axis/.append style={width=5.3cm}}
\begin{tikzpicture}
\begin{loglogaxis}
\addplot coordinates
    {(10,1e-5) (20,5e-6) (40,2.5e-6)};
\end{loglogaxis}
\end{tikzpicture}

\begin{tikzpicture}
\begin{loglogaxis}[
    extra x ticks={20,40},
    extra y ticks={5e-6,2.5e-6}]
\addplot coordinates
    {(10,1e-5) (20,5e-6) (40,2.5e-6)};
\end{loglogaxis}
\end{tikzpicture}

\begin{tikzpicture}
\begin{loglogaxis}[
    log identify minor tick positions=false,
    extra x ticks={20,40},
    extra y ticks={5e-6,2.5e-6}]
\addplot coordinates
    {(10,1e-5) (20,5e-6) (40,2.5e-6)};
\end{loglogaxis}
\end{tikzpicture}
```

Remarks:

- Use `extra x ticks` to highlight special tick positions. The use of `extra x ticks` does not affect minor tick/grid line generation, so you can place extra ticks at positions $j \cdot 10^i$ in log–plots.
- Extra ticks are always typeset as major ticks.
  They are affected by `major tick length` or options like `grid=major`.
- Use the style `every extra x tick` (`every extra y tick`) to configure the appearance.
- You can also use '`extra x tick style={`⟨...⟩`}`' which has the same effect.

`/pgfplots/space between ticks=`{⟨*number*⟩} (no default, initially `35`)
`/pgfplots/try min ticks=`{⟨*number*⟩} (no default, initially `4`)
`/pgfplots/try min ticks log=`{⟨*number*⟩} (no default, initially `3`)

see Options `xtick` and `ytick` for a description.

`/pgfplots/tickwidth=`{⟨*dimension*⟩} (no default, initially `0.15cm`)
`/pgfplots/major tick length=`{⟨*dimension*⟩} (no default, initially `0.15cm`)

Sets the width of major tick lines.

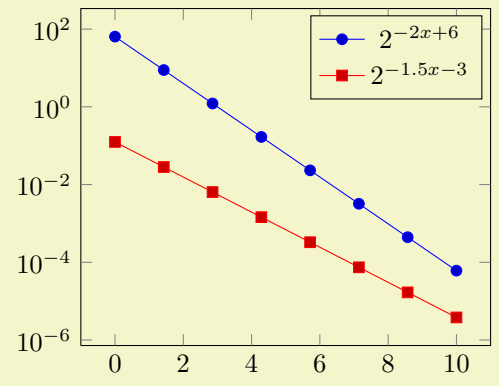
`/pgfplots/subtickwidth=`{⟨*dimension*⟩} (no default, initially `0.1cm`)
`/pgfplots/minor tick length=`{⟨*dimension*⟩} (no default, initially `0.1cm`)

Sets the width of minor tick lines.

`/pgfplots/xtickten=`{⟨*exponent base 10 list*⟩} (no default)
`/pgfplots/ytickten=`{⟨*exponent base 10 list*⟩} (no default)

These options allow to place ticks at selected positions $10^k, k \in$ {⟨*exponent base 10 list*⟩}. They are only used for logplots. The syntax for {⟨*exponent base 10 list*⟩} is the same as above for `xtick=`{⟨*list*⟩} or `ytick=`{⟨*list*⟩}.

Using '`xtickten={1,2,3,4}`' is equivalent to '`xtick={1e1,1e2,1e3,1e4}`', but it requires fewer computational time and it allows to use the short syntax '`xtickten={1,...,4}`'.




```
\begin{tikzpicture}[samples=8]
\begin{semilogyaxis}[
    ytickten={-6,-4,...,4},
    domain=0:10]

% invoke gnuplot to generate coordinates:
\addplot plot[id=pow1]
    function {2**(-2*x + 6)};
\addlegendentry{$2^{-2x + 6}$}

\addplot plot[id=pow2]
    function {2**(-1.5*x -3)};
\addlegendentry{$2^{-1.5x -3}$}
\end{semilogyaxis}
\end{tikzpicture}
```

/pgfplots/xticklabels={⟨*label list*⟩} (no default)
/pgfplots/yticklabels={⟨*label list*⟩} (no default)

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

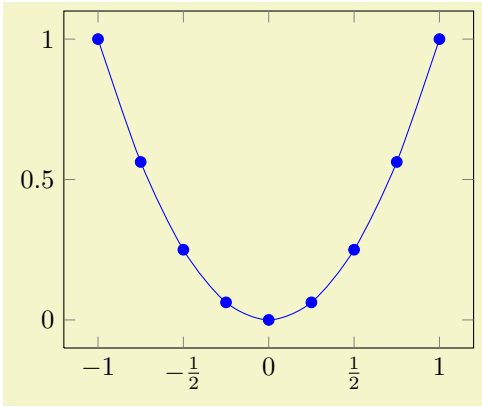
This is one of two options to assign tick labels directly. The other option is `xticklabel={`⟨*command*⟩`}` (or `yticklabel={`⟨*command*⟩`}`). Option '`xticklabel`' offers higher flexibility while '`xticklabels`' is easier to use.

The argument `{`⟨*label 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}, $e$}.
```




```
\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=*] 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}
```

/pgfplots/xticklabel={⟨*command*⟩} (no default)
/pgfplots/yticklabel={⟨*command*⟩} (no default)

Use `xticklabel` or `yticklabel` to change the TEX-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={`⟨*label list*⟩`}`' (or `yticklabels={`⟨*label list*⟩`}`). Option '`xticklabel`' offers higher flexibility while '`xticklabels`' is easier to use.

The argument {⟨*command*⟩} can be any TEX-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).

\nexttick This command is only valid in case if the `x tick label as interval` option is set (or the corresponding variable for y). It will contain the position of the next tick position, that means the right boundary of the tick interval.

The default argument is

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

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

```
\def\axisdefaultticklabel{%
    $\pgfmathprintnumber{\tick}$%
}

\def\axisdefaultticklabellog{%
    \pgfkeysgetvalue{/pgfplots/log number format code/.@cmd}\pgfplots@log@label@style
    \expandafter\pgfplots@log@label@style\tick\pgfeov
}
```

that means you can configure the appearance of linear axis with the number formatting options described in section 7.7 and logarithmic axis with `log number format code`, see below.

You can change the appearance of tick labels with

```
\pgfplotsset{every tick label/.append style={
    font=\tiny,
    /pgf/number format/sci}}
```

and/or

```
\pgfplotsset{every x tick label/.append style={
    above,
    /pgf/number format/fixed zerofill}}
```

and

```
\pgfplotsset{every y tick label/.append style={font=\bfseries}}
```

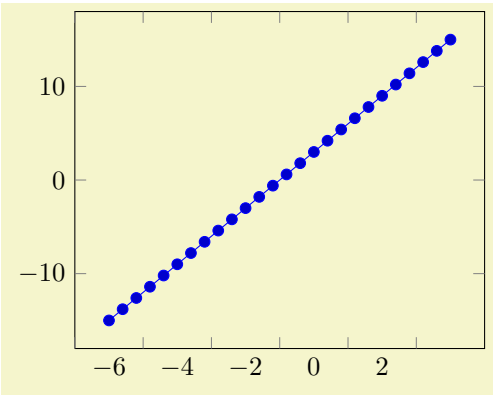
Another possibility is to use

```
\begin{axis}[y tick label style={above,
    /pgf/number format/fixed zerofill}
]
...
\end{axis}
```

which has the same effect as the '`every x tick label`' statement above. This is possible for all PGFPLOTS-`every`-styles, see section 7.10.

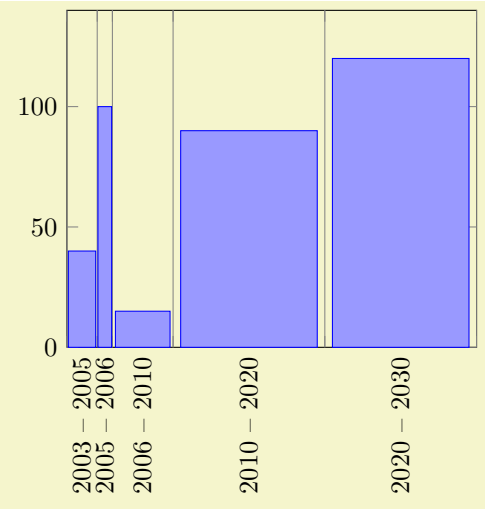
`/pgfplots/x tick label as interval`=`true`|`false` (no default, initially `false`)
`/pgfplots/y tick label as interval`=`true`|`false` (no default, initially `false`)

Allows to treat tick labels as intervals; that means the tick positions denote the interval boundaries. If there are n positions, $(n-1)$ tick labels will be generated, one for each interval.




```
\begin{tikzpicture}
\begin{axis}[x tick label as interval]
    \addplot (\x,3*\x);
\end{axis}
\end{tikzpicture}
```

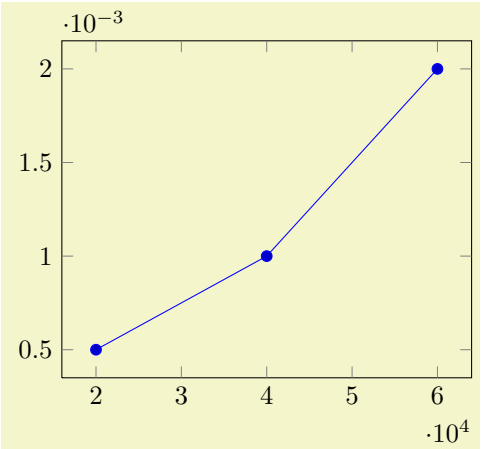
This mode enables the use of `\nexttick` inside of `xticklabel` (or `yticklabel`). A common application might be a bar plot.



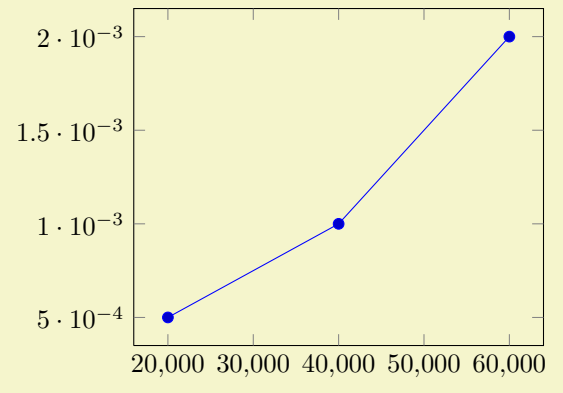

```
\begin{tikzpicture}
\begin{axis}[
    ybar interval=0.9,
    x tick label as interval,
    xmin=2003,xmax=2030,
    ymin=0,ymax=140,
    xticklabel={
       $\pgfmathprintnumber{\tick}$
    -- $\pgfmathprintnumber{\nexttick}$},
    xtick=data,
    x tick label style={
        rotate=90,anchor=east,
        /pgf/number format/1000 sep=}
]

    \addplot[draw=blue,fill=blue!40!white]
        coordinates
        {(2003,40) (2005,100) (2006,15)
         (2010,90) (2020,120) (2030,3)};
\end{axis}
\end{tikzpicture}
```

/pgfplots/scaled ticks=true|false (no default, initially true)

Allows to factor out common exponents in tick labels. For example, if you have tick labels 20000, 40000 and 60000, you may want to save some space and write 2, 4, 6 with a separate factor '$\cdot 10^4$'. Use 'scaled ticks=true' to enable this feature (default is true).




```
\begin{tikzpicture}
\begin{axis}[scaled ticks=true]
    \addplot coordinates {
        (20000,0.0005)
        (40000,0.0010)
        (60000,0.0020)
    };
\end{axis}
\end{tikzpicture}%
```




```
\begin{tikzpicture}
\begin{axis}[scaled ticks=false]
    \addplot coordinates {
        (20000,0.0005)
        (40000,0.0010)
        (60000,0.0020)
    };
\end{axis}
\end{tikzpicture}
```

/pgfplots/tick scale label code/.code={⟨...⟩}

Allows to change the default code for scaled tick labels. The default is

```
tick scale label code/.code={$\cdot 10^{#1}$}.
```

/pgfplots/scale ticks below={⟨*exponent*⟩} (no default)

Allows fine tuning of the 'scaled ticks' algorithm: if the axis limits are of magnitude 10^e and $e <$ {⟨*exponent*⟩}, the common prefactor 10^e will be factored out. The default is -1.

/pgfplots/scale ticks above={⟨*exponent*⟩} (no default)

Allows fine tuning of the 'scaled ticks' algorithm: if the axis limits are of magnitude 10^e and $e >$ {⟨*exponent*⟩}, the common prefactor 10^e will be factored out. The default is 3.

/pgfplots/tickpos=left|right|both (no default, initially 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.

/pgfplots/tick align=inside|center|outside (no default, initially inside)

Allows to change the location of the ticks relative to the axis lines. Default is "inside". this setting applies to both x and y axis.

/pgfplots/xminorticks=true|false (no default, initially true)
/pgfplots/yminorticks=true|false (no default, initially true)
/pgfplots/xmajorticks=true|false (no default, initially true)
/pgfplots/ymajorticks=true|false (no default, initially true)
/pgfplots/ticks=minor|major|both|none (no default, initially both)

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 between tick positions. Please note that minor ticks are automatically disabled if xtick is not a uniform range[7].

The key minor tick length={⟨*dimen*⟩} configures the tick length for minor ticks while the major variant applies to major ticks. You can configure the appearance using the following styles:

```
\pgfplotsset{every tick/.append style={color=black}} % applies to major and minor ticks,
\pgfplotsset{every minor tick/.append style={thin}}  % applies only to minor ticks,
\pgfplotsset{every major tick/.append style={thick}} % applies only to major ticks.
```

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

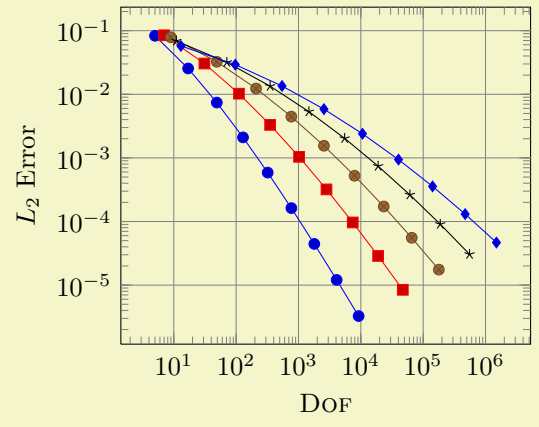
/pgfplots/xminorgrids=true|false (no default, initially true)
/pgfplots/yminorgrids=true|false (no default, initially true)
/pgfplots/xmajorgrids=true|false (no default, initially true)
/pgfplots/ymajorgrids=true|false (no default, initially true)

[7] A uniform list means the difference between all elements is the same for linear axis or, for logarithmic axes, log(10).

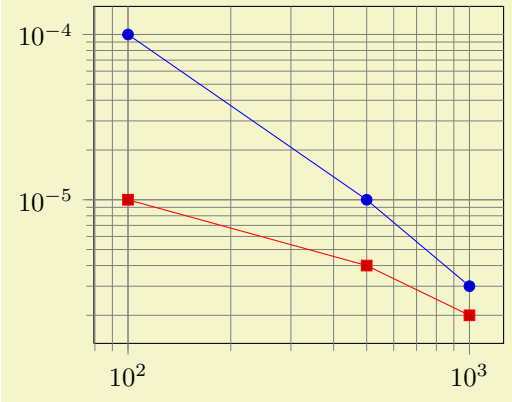
/pgfplots/grids=minor|major|both|none (no default, initially both)

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).

This example employs the coordinates defined on page 8.




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




```
\begin{tikzpicture}
\begin{loglogaxis}[
    grid=both,
    tick align=outside,
    tickpos=left]
\addplot coordinates
    {(100,1e-4) (500,1e-5) (1000,3e-6)};
\addplot coordinates
    {(100,1e-5) (500,4e-6) (1000,2e-6)};
\end{loglogaxis}
\end{tikzpicture}
```

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

```
\pgfplotsset{every axis grid/.style={style=help lines}}
\pgfplotsset{every minor grid/.append style={color=blue}}
\pgfplotsset{every major grid/.append style={thick}}
```

7.10 Style Options

7.10.1 All Supported Styles

PGFPLOTS provides many styles to customize its appearance and behavior. They can be defined and changed in any place where keys are allowed.

Key handler ⟨*key*⟩/.style={⟨*key-value-list*⟩}

Defines or redefines a style ⟨*key*⟩. A style is a normal key which will set all options in {⟨*key-value-list*⟩} when it is set.

Use \pgfplotsset{⟨*key*⟩/.style={⟨*key-value-list*⟩}} to (re-) define a style ⟨*key*⟩ in the namespace /pgfplots.

Key handler ⟨*key*⟩/.append style={⟨*key-value-list*⟩}

Appends {⟨*key-value-list*⟩} to an already existing style ⟨*key*⟩. This is the preferred method to change the predefined styles: if you only append, you maintain compatibility with future versions.

Use \pgfplotsset{⟨*key*⟩/.append style={⟨*key-value-list*⟩}} to append {⟨*key-value-list*⟩} to the style ⟨*key*⟩. This will assume the prefix /pgfplots.

\pgfplotsset{⟨*key-value-list*⟩}

Defines or sets all options in {⟨*key-value-list*⟩}.

It is a shortcut for \pgfqkeys{/pgfplots}{⟨*key-value-list*⟩}, that means it inserts the prefix /pgfplots to any option which has no full path.

You can define new style keys with .style and .append style, see above.

/pgfplots/every axis (style, no value)

Installed at the beginning of every axis. TikZ options inside of it will be used for anything inside of the axis rectangle and any axis descriptions.

/pgfplots/every semilogx axis (style, no value)

Installed at the beginning of every plot with linear x axis and logarithmic y axis, but after 'every axis'.

/pgfplots/every semilogy axis (style, no value)

Likewise, but with interchanged roles for x and y.

/pgfplots/every loglog axis (style, no value)

Installed at the beginning of every double–logarithmic plot.

/pgfplots/every linear axis (style, no value)

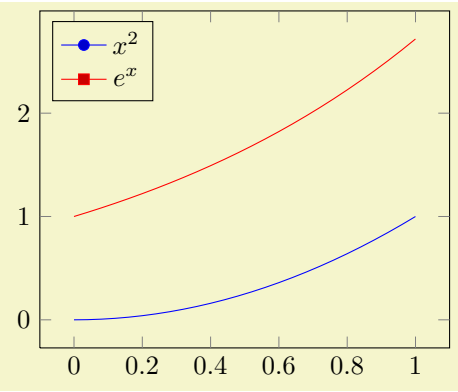
Installed at the beginning of every plot with normal axis scaling.

/pgfplots/every axis plot (style, no value)

Used for the TikZ-drawing command in any '\addplot' command. May only contain TikZ options.

/pgfplots/every axis plot post (style, no value)

This style is very similar to every axis plot in that is applies to any drawing command in \addplot. However, it is set *after* any user defined styles or cycle list options.




```
\begin{tikzpicture}
\pgfplotsset{
    every axis plot post/.append style=
        {mark=none}}

\begin{axis}[
    legend style={
        at={(0.03,0.97)},anchor=north west},
    domain=0:1]
    \addplot (\x,\x^2);
    \addplot (\x,{exp(\x)});
    \legend{$x^2$,$e^x$}
\end{axis}
\end{tikzpicture}
```

/pgfplots/every axis plot no # (style, no value)

Used for every #th plot where $\# = 1, 2, 3, 4, \ldots$.

/pgfplots/every axis label (style, no value)

Used for x and y axis label. You can use 'at={⟨*(x,y)*⟩} to set its position where $(0, 0)$ refers to the lower left corner and $(1, 1)$ to the upper right one.

/pgfplots/every axis x label (style, no value)

Used for x labels, installed after 'every axis label'.

/pgfplots/every axis y label (style, no value)

Like 'every axis x label', just for y.

/pgfplots/every axis title (style, no value)

Used for any axis title. The at={⟨*(x,y)*⟩} command works as for 'every axis label'.

/pgfplots/every tick (style, no value)

Installed for each of the small tick lines.

/pgfplots/every minor tick (style, no value)

Used for each minor tick line, installed after ‘every tick’.

/pgfplots/every major tick (style, no value)

Used for each major tick line, installed after ‘every tick’.

/pgfplots/every x tick (style, no value)

/pgfplots/every minor x tick (style, no value)

/pgfplots/every major x tick (style, no value)

/pgfplots/every y tick (style, no value)

/pgfplots/every minor y tick (style, no value)

/pgfplots/every major y tick (style, no value)

/pgfplots/every axis grid (style, no value)

Used for each grid line.

/pgfplots/every minor grid (style, no value)

Used for each minor grid line, installed after ‘every axis grid’.

/pgfplots/every major grid (style, no value)

Likewise, for major grid lines.

/pgfplots/every axis x grid (style, no value)

/pgfplots/every minor x grid (style, no value)

/pgfplots/every major x grid (style, no value)

/pgfplots/every axis y grid (style, no value)

/pgfplots/every minor y grid (style, no value)

/pgfplots/every major y grid (style, no value)

/pgfplots/every tick label (style, no value)

Used for each x and y tick labels.

/pgfplots/every x tick label (style, no value)

Used for each x tick label, installed after ‘every tick label’.

/pgfplots/every y tick label (style, no value)

Likewise, for y tick labels.

/pgfplots/every x tick scale label (style, no value)

Configures placement and display of the nodes containing the order of magnitude of x tick labels, see 7.9 for more information about scaled ticks.

/pgfplots/every y tick scale label (style, no value)

Likewise, but for y-tick scale labels.

/pgfplots/every extra x tick (style, no value)

Allows to configure the appearance of ‘extra x ticks’. This style is installed before touching the first extra x tick, so you can set any option which affects tick generation, for example

/pgfplots/every error bar (style, no value)

Installed for every error bar.

```
\pgfplotsset{every extra x tick/.append style={grid=major}}
\pgfplotsset{every extra x tick/.append style={major tick length=0pt}}
\pgfplotsset{every extra x tick/.append style={/pgf/number format=sci subscript}}
```

or something like that.

/pgfplots/every extra y tick (style, no value)

Likewise, but for extra y-ticks.

/pgfplots/every axis legend (style, no value)

Installed for each legend. As for `every axis label`, the legend's position can be placed using coordinates between 0 and 1, see above.

/pgfplots/x tick label style={⟨...⟩} (no default)
/pgfplots/y tick label style={⟨...⟩} (no default)
/pgfplots/x tick scale label style={⟨...⟩} (no default)
/pgfplots/y tick scale label style={⟨...⟩} (no default)
/pgfplots/label style={⟨...⟩} (no default)
/pgfplots/x label style={⟨...⟩} (no default)
/pgfplots/y label style={⟨...⟩} (no default)
/pgfplots/title style={⟨...⟩} (no default)
/pgfplots/tick style={⟨...⟩} (no default)
/pgfplots/minor tick style={⟨...⟩} (no default)
/pgfplots/major tick style={⟨...⟩} (no default)
/pgfplots/x tick style={⟨...⟩} (no default)
/pgfplots/y tick style={⟨...⟩} (no default)
/pgfplots/minor x tick style={⟨...⟩} (no default)
/pgfplots/minor y tick style={⟨...⟩} (no default)
/pgfplots/major x tick style={⟨...⟩} (no default)
/pgfplots/major y tick style={⟨...⟩} (no default)
/pgfplots/grid style={⟨...⟩} (no default)
/pgfplots/minor grid style={⟨...⟩} (no default)
/pgfplots/major grid style={⟨...⟩} (no default)
/pgfplots/x grid style={⟨...⟩} (no default)
/pgfplots/y grid style={⟨...⟩} (no default)
/pgfplots/minor x grid style={⟨...⟩} (no default)
/pgfplots/minor y grid style={⟨...⟩} (no default)
/pgfplots/major x grid style={⟨...⟩} (no default)
/pgfplots/major y grid style={⟨...⟩} (no default)
/pgfplots/extra x tick style={⟨...⟩} (no default)
/pgfplots/extra y tick style={⟨...⟩} (no default)

All these options are equivalent to the corresponding '`every ...`'–styles.

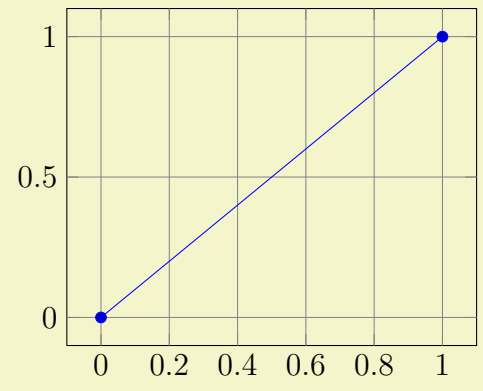
For example, `label style={⟨...⟩}` has the same effect as

```
\pgfplotsset{every axis label/.append style={...}}
```

but can be provided as an option (or as part of a user defined style). See section 7.10 for more information about the available styles.

7.10.2 Assigning Own Styles

Use `\pgfplotsset{⟨`*style name*`⟩/.style={⟨`*key-value-list*`⟩}}` to create own styles. You *can't use* `\tikzstyle{⟨`*style name*`⟩}=[]`.




```
\pgfplotsset{my personal style/.style=
    {grid=major,font=\large}}

\begin{tikzpicture}
\begin{axis}[my personal style]
    \addplot coordinates {(0,0) (1,1)};
\end{axis}
\end{tikzpicture}
```

7.11 Alignment Options

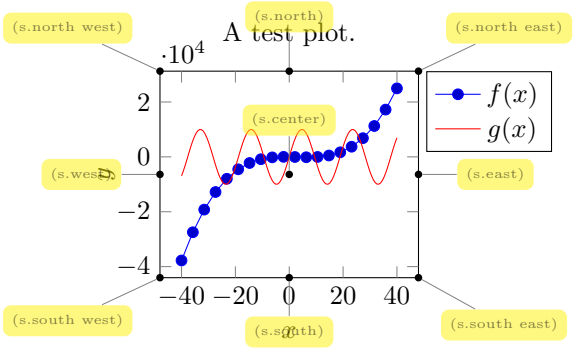
/pgfplots/anchor={⟨*name*⟩} (no default, initially `south west`)

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

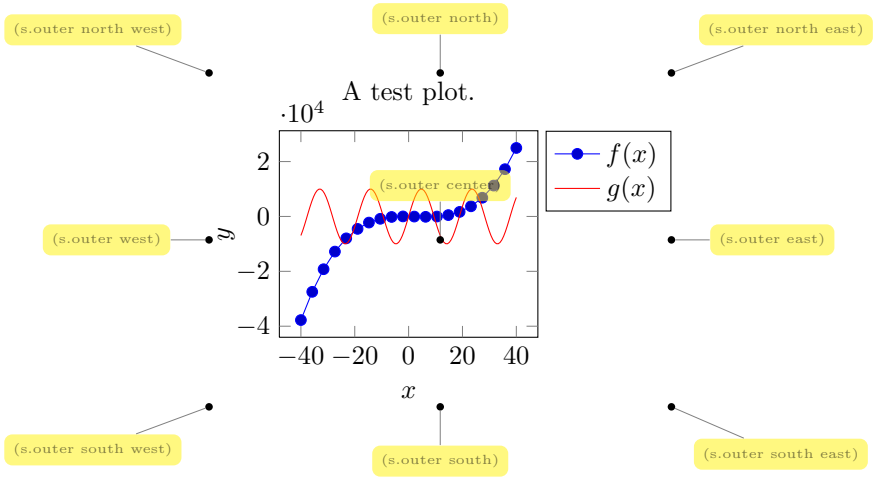
Anchors are useful in conjunction with horizontal or vertical alignment of plots, see the examples below.

There are three sets of anchors available: anchors positioned on the axis rectangle, anchors on the outer bounding box and anchors which have one coordinate on the outer bounding box and the other one at a position of the axis rectangle.

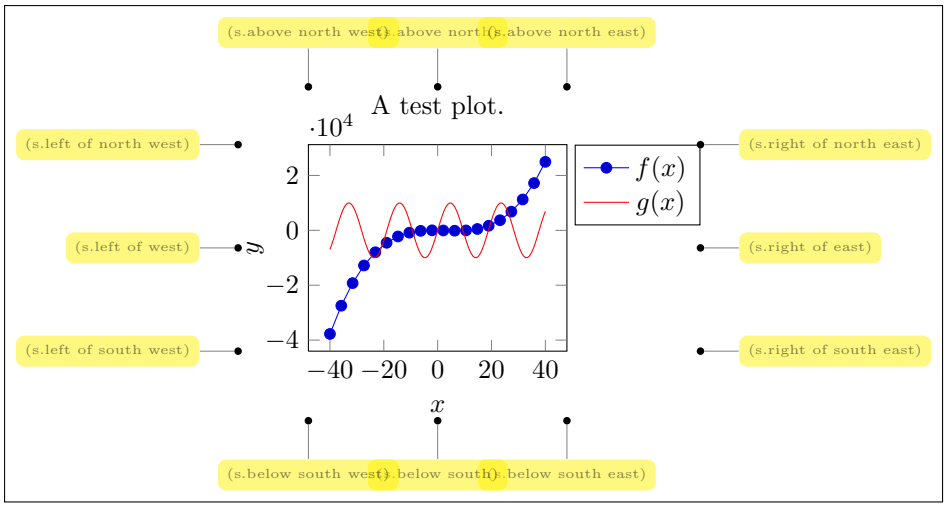
In more detail, we have Anchors on the axis rectangle,



Anchors on the outer bounding box,

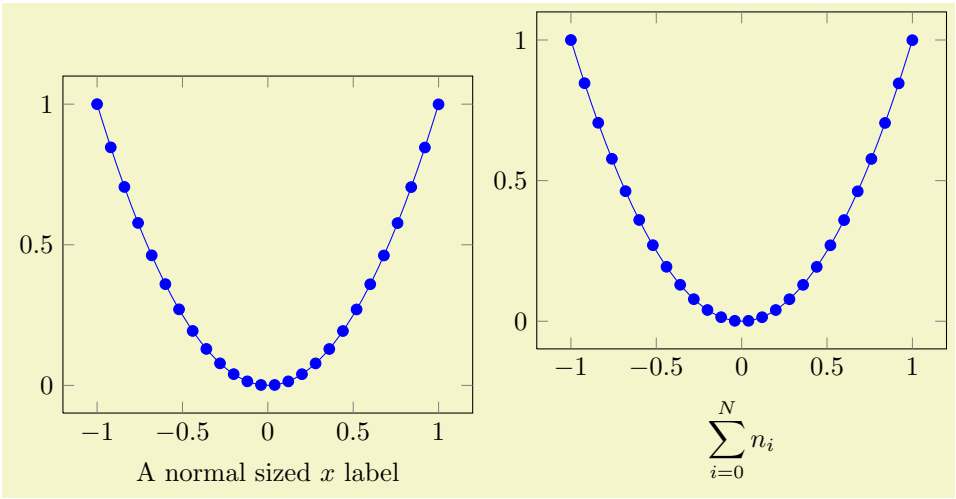


Finally there are anchors which have one coordinate on the outer bounding box, and one on the axis rectangle,

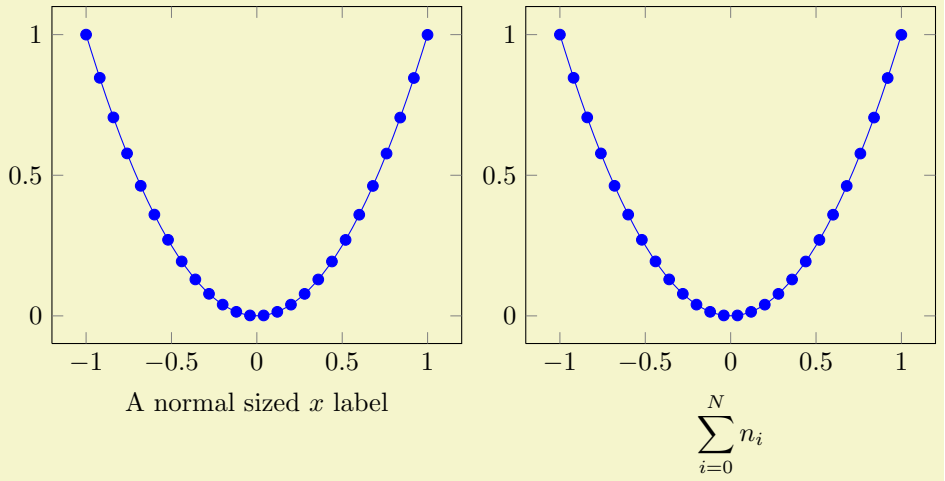


The default value is `anchor=south west`. You can use anchors in conjunction with the Ti*k*Z `baseline` option and/or `\begin{pgfinterruptboundingbox}` to perform alignment.

Vertical alignment with `baseline` 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:



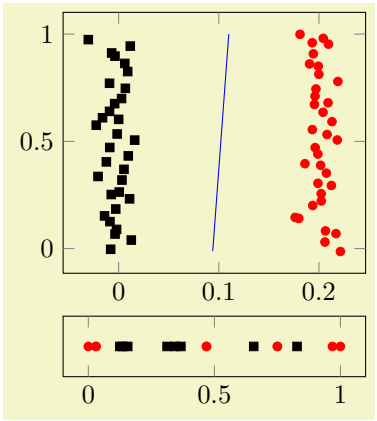

```
\tikzset{domain=-1:1}
\begin{tikzpicture}
    \begin{axis}[xlabel=A normal sized $x$ label]
    \addplot[smooth,blue,mark=*] (\x,\x^2);
    \end{axis}
\end{tikzpicture}%
\hspace{0.15cm}
\begin{tikzpicture}
    \begin{axis}[xlabel={$\displaystyle \sum_{i=0}^N n_i $ }]
    \addplot[smooth,blue,mark=*] (\x,\x^2);
    \end{axis}
\end{tikzpicture}
```




```
\tikzset{domain=-1:1}
\begin{tikzpicture}[baseline]
    \begin{axis}[xlabel=A normal sized $x$ label]
    \addplot[smooth,blue,mark=*] (\x,\x^2);
    \end{axis}
\end{tikzpicture}%
\hspace{0.15cm}
\begin{tikzpicture}[baseline]
    \begin{axis}[xlabel={$\displaystyle \sum_{i=0}^N n_i $ }]
    \addplot[smooth,blue,mark=*] (\x,\x^2);
    \end{axis}
\end{tikzpicture}
```

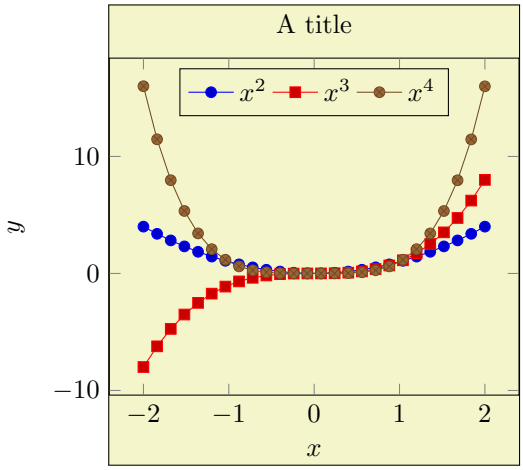
The `baseline` option configures Ti*k*Z to shift position $y = 0$ to the text's baseline and the `south west` anchor shifts the axis such the $y = 0$ is at the lower left axis corner.

Horizontal Alignment If you place multiple `axes` into a single `tikzpicture` and use the '`anchor`'-option, you can control horizontal alignment:




```
\begin{tikzpicture}
\pgfplotsset{every axis/.append style={
cycle list={
    {red,only marks,mark options={
        fill=red,scale=0.8},mark=*},
    {black,only marks,mark options={
        fill=black,scale=0.8},mark=square*}}}}

\begin{axis}[width=4cm,scale only axis,
    name=main plot]
\addplot file
    {plotdata/pgfplots_scatterdata1.dat};
\addplot file
    {plotdata/pgfplots_scatterdata2.dat};
\addplot[blue] coordinates {
    (0.093947,    -0.011481)
    (0.101957,    0.494273)
    (0.109967,    1.000027)};
\end{axis}

% introduce named coordinate:
\path (main plot.below south west) ++(0,-0.1cm)
    coordinate (lower plot position);

\begin{axis}[at={(lower plot position)},
    anchor=north west,
    width=4cm,scale only axis,height=0.8cm,
    ytick=\empty]

\addplot file
  {plotdata/pgfplots_scatterdata1_latent.dat};
\addplot file
  {plotdata/pgfplots_scatterdata2_latent.dat};
\end{axis}
\end{tikzpicture}
```

Bounding box restrictions The following figure is centered and encapsulated with an **\fbox** to show its bounding box.




```
\fbox{%
\begin{tikzpicture}%
    \begin{pgfinterruptboundingbox}
    \begin{axis}[
        name=my plot,
        title=A title,
        xlabel={$x$},
        ylabel={$y$},
        legend style={at={(0.5,0.97)},
            anchor=north,legend columns=-1},
        domain=-2:2
    ]
    \addplot (\x,\x^2);
    \addplot (\x,\x^3);
    \addplot (\x,\x^4);
    \legend{$x^2$,$x^3$,$x^4$}
    \end{axis}
    \end{pgfinterruptboundingbox}

    \useasboundingbox
              (my plot.below south west)
    rectangle (my plot.above north east);
\end{tikzpicture}%
}%
```

The `pgfinterruptboundingbox` environment does not include its content into the image's bounding box, and `\useasboundingbox` sets the pictures bounding box to the following argument.

`/pgfplots/at={`⟨*coordinate expression*⟩`}` (no default)

Assigns a position for the complete axis image. This option works similarly to the `at`-option of `\node[at={`⟨*coordinate expression*⟩`}]`, see [1]. The common syntax is `at={`⟨*(x,y)*⟩`}`.

7.12 Miscellaneous Options

`/pgfplots/disablelogfilter=true|false` (initally false, default true) (no default)

Disables numerical evaluation of $\log(x)$ in TEX. 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.

`/pgfplots/disabledatascaling=true|false` (initally false, default true) (no default)

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

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

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 magnitude* 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. This scaling allows precision beyond TEX's capabilities.

The option "`disabledatascaling`" disables this data transformation. That means you are restricted to coordinates which TEX can handle[8].

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

[8] 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.

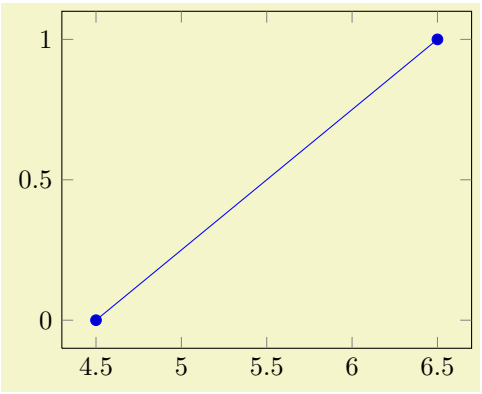
/pgfplots/x filter/.code={⟨...⟩} (no default)
/pgfplots/y filter/.code={⟨...⟩} (no default)

The code keys `x filter` and `y filter` allow coordinate filtering. A coordinate filter gets an input coordinate as `#1`, applies some operation and writes the result into the macro `\pgfmathresult`. If `\pgfmathresult` is empty afterwards, the coordinate is discarded.

It is allowed if filters do not `\pgfmathresult`. In this case, the unfiltered coordinate will be used.

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.

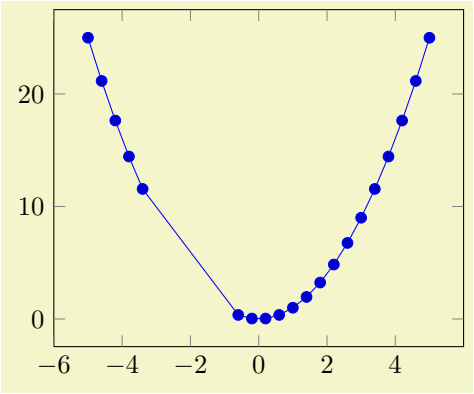
The following filter adds 0.5 to every x coordinate.




```
\begin{tikzpicture}
\begin{axis}[x filter/.code=
    {\pgfmathadd{#1}{0.5}}]
\addplot coordinates {
    (4,0)
    (6,1)
};
\end{axis}
\end{tikzpicture}
```

Please refer to [1, pgfmath manual] for details about the math engine of PGF. Please keep in mind that the math engine works with limited TEX precision.

During evaluation of the filter, the macro `\coordindex` contains the number of the current coordinate (starting with 0). Thus, the following filter discards all coordinates after the 5th and before the 10th.




```
\begin{tikzpicture}
\begin{axis}[
    samples=20,
    x filter/.code={
        \ifnum\coordindex>4\relax
            \ifnum\coordindex<11\relax
                \def\pgfmathresult{}
            \fi
        \fi
    }]
\addplot (\x,\x^2);
\end{axis}
\end{tikzpicture}
```

There is also a style key which simplifies selection by index, see below.

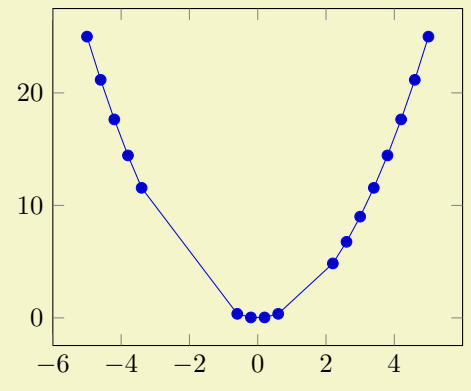
PGFPLOTS invokes the filter with argument `#1` set to the input coordinate. For x-filters, this is the x-coordinate as it is specified to `\addplot`, for y-filters it is the y-coordinate.

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.

/pgfplots/skip coords between index={⟨*begin*⟩}{⟨*end*⟩} (style, no default)

A style which appends an `x filter` which discards selected coordinates. The selection is done by index where indexing starts with 0, see `\coordindex`. Every coordinate with index $\langle begin\rangle \le i < \langle end\rangle$ will be skipped.




```
\begin{tikzpicture}
\begin{axis}[
    samples=20,
    skip coords between index={5}{11},
    skip coords between index={15}{18}]

\addplot (\x,\x^2);
\end{axis}
\end{tikzpicture}
```

/pgfplots/filter discard warning=true|false (no default, initially true)

Issues a notification in your logfile whenever coordinate filters discard coordinates.

/pgfplots/execute at begin plot={⟨*commands*⟩} (no default)

This axis option allows to invoke {⟨*commands*⟩} at the beginning of each \addplot command. The argument {⟨*commands*⟩} can be any TEX content.

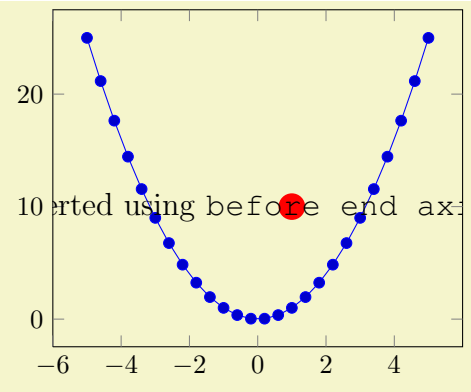
You may use this in conjunction with xfilter=... to reset any counters or whatever. An example would be to change every 4th coordinate.

/pgfplots/execute at end plot={⟨*commands*⟩} (no default)

This axis option allows to invoke {⟨*commands*⟩} after each \addplot command. The argument {⟨*commands*⟩} can be any TEX content.

/pgfplots/before end axis/.code={⟨...⟩}

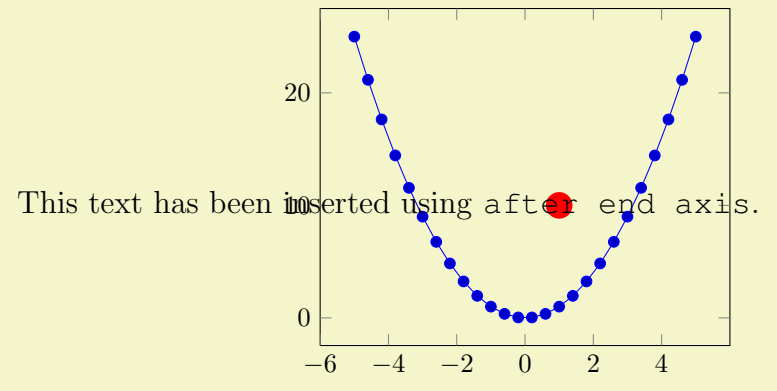
Allows to insert {⟨*commands*⟩} just before the axis is ended. This option takes effect inside of the clipped area.




```
\pgfplotsset{every axis/.append style={
    before end axis/.code={
        \fill[red] (axis cs:1,10) circle(5pt);
        \node at (axis cs:-4,10)
            {\large This text has been inserted
             using \texttt{before end axis}.};
    }}}
\begin{tikzpicture}
    \begin{axis}
    \addplot (\x,\x^2);
    \end{axis}
\end{tikzpicture}
```

/pgfplots/after end axis/.code={⟨...⟩}

Allows to insert {⟨*commands*⟩} right after the end of the clipped drawing commands. While befor end axis has the same effect as if {⟨*commands*⟩} had been placed inside of your axis, after end axis allows to access axis coordinates without being clipped.

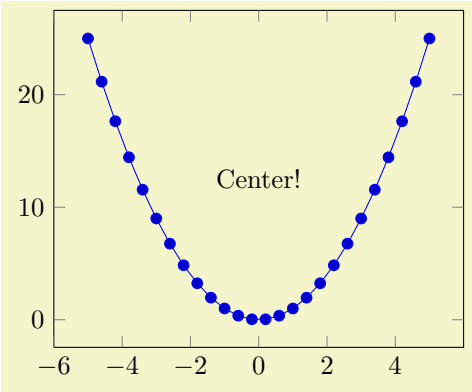



```
\pgfplotsset{every axis/.append style={
    after end axis/.code={
        \fill[red] (axis cs:1,10) circle(5pt);
        \node at (axis cs:-4,10)
            {\large This text has been inserted using \texttt{after end axis}.};
    }}}
\begin{tikzpicture}
    \begin{axis}
    \addplot (\x,\x^2);
    \end{axis}
\end{tikzpicture}
```

/pgfplots/extra description/.code={⟨...⟩}

Allows to insert {⟨*commands*⟩} after axis labels, titles and legends have been typeset.

The code can thus use $(0, 0)$ to access the lower left corner and $(1, 1)$ to access the upper right one. It won't be clipped.




```
\pgfplotsset{every axis/.append style={
    extra description/.code={
        \node at (0.5,0.5) {Center!};
    }}}
\begin{tikzpicture}
    \begin{axis}
    \addplot (\x,\x^2);
    \end{axis}
\end{tikzpicture}
```

8 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.

8.1 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.

8.2 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.

8.3 matlab2pgfplots.sh

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

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

See matlab2pgfplots.m above.

Index

References

[1] T. Tantau. Ti*k*Z and PGF manual. `http://sourceforge.net/projects/pgf`. $v. \geq 2.00$.