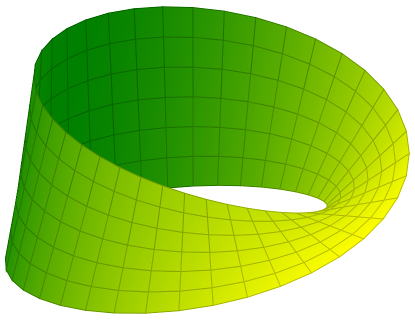
Quick Reference

PGFPLOTS

```
\begin{tikzpicture}
\begin{axis}[
  hide axis,
  view = {40}{40},
]
\addplot3[
  surf,
  colormap/greenyellow,
  shader = faceted interp,
  z buffer = sort,
  point meta = x,
  domain = 0:360,
  domain y = -0.5:0.5,
  samples = 40,
  samples y = 7,
]
({(1 + 0.5 * y * cos(x / 2))) * cos(x)},
 {(1 + 0.5 * y * cos(x / 2))) * sin(x)},
 {0.5 * y * sin(x/2)});
\end{axis}
\end{tikzpicture}
```



Ralph Schleicher

Contents

Nomenclature

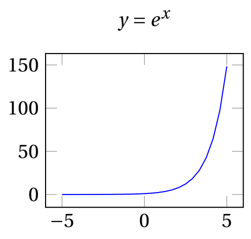
`\foo`	TeX control sequence.
`foo`$_{\text{env}}$	LaTeX environment `foo`.
`foo`$_{\text{sty}}$	PGFPLOTS style with key `foo`.
`foo`	Terminal symbol, literal text.
$\langle foo\rangle$	Non-terminal symbol, metasyntactic variable.
$\langle foo\rangle \to \langle bar\rangle$	Production rule; $\langle foo\rangle$ can be replaced by $\langle bar\rangle$, $\langle foo\rangle$ and $\langle bar\rangle$ are implicit groups.
$\langle foo\rangle\ \langle bar\rangle$	Sequence; $\langle foo\rangle$ followed by $\langle bar\rangle$.
$\langle foo\rangle \mid \langle bar\rangle$	Choice; $\langle foo\rangle$ or $\langle bar\rangle$.
$\langle foo\rangle^*$	$\langle foo\rangle$ can occur zero or more times.
$\langle foo\rangle^+$	$\langle foo\rangle$ can occur one or more times.
$\langle foo\rangle^?$	$\langle foo\rangle$ is optional.
( )	Explicit group.
▷ $\langle key\rangle$ = $\langle value\rangle$	User option, $\langle key\rangle$ and $\langle value\rangle$ are implicit groups.
▷ $\langle key\rangle$	User option without a value.
$\underline{42}$	Default value is 42.
↲	Line continuation mark.
$\langle empty\rangle$	Nothing.
$\langle newline\rangle$	Newline character, `^^M` in TeX.
$\langle dimension\rangle$	A legitimate TeX dimension.
$\langle number\rangle$	$(-\infty,\infty)\cap\mathbb{R}$.
$\langle positive\ number\rangle$	$(0,\infty)\cap\mathbb{R}$.
$\langle non\text{-}negative\ number\rangle$	$[0,\infty)\cap\mathbb{R}$.
$\langle integer\rangle$	$(-\infty,\infty)\cap\mathbb{Z}$.
$\langle positive\ integer\rangle$	$(0,\infty)\cap\mathbb{Z}$.
$\langle non\text{-}negative\ integer\rangle$	$[0,\infty)\cap\mathbb{Z}$.

1 General

1.1 Document Structure

```
\documentclass{standalone}
\usepackage{pgfplots}
\pgfplotsset{compat=1.16}

\begin{document}
\begin{tikzpicture}
\begin{axis}[title={$y = e^x$}]
\addplot+[no markers] {exp(x)};
\end{axis}
\end{tikzpicture}
\end{document}
```



1.2 PGFPLOTS Options

`\pgfplotsset{`⟨*key/value list*⟩`}`
⟨*key/value list*⟩ → (⟨*key*⟩ `=` ⟨*value*⟩`,`)*

Options are supplied as a ⟨*key/value list*⟩. The `/pgfplots/` and `/tikz/` prefixes in ⟨*key*⟩ can be omitted in the scope of PGFPLOTS commands. Please note that a trailing comma in ⟨*key/value list*⟩ does no harm.

1.3 Key Handlers

`\pgfplotsset{`⟨*key*⟩`/.style = {`⟨*key/value list*⟩`}}`
Define or replace style ⟨*key*⟩.

`\pgfplotsset{`⟨*key*⟩`/.append style = {`⟨*key/value list*⟩`}}`
Append to style ⟨*key*⟩.

`\pgfplotsset{`⟨*key*⟩`/.code = {`⟨*TEX code*⟩`}}`
Define or replace ⟨*key*⟩ that – when run – takes one argument; ⟨*TEX code*⟩ can refer to the supplied argument as `#1`. Invoke as '`\pgfplotsset{`⟨*key*⟩ `= {`⟨*argument*⟩`}}`'.

`\pgfplotsset{`⟨*key*⟩`/.code 2 args = {`⟨*TEX code*⟩`}}`
Like ⟨*key*⟩`/.code` but with two arguments; ⟨*TEX code*⟩ can refer to the supplied arguments as `#1` and `#2`. Invoke as '`\pgfplotsset{`⟨*key*⟩ `= {`⟨*first argument*⟩`}{`⟨*second argument*⟩`}}`'.

`\pgfplotsset{`⟨*key*⟩`/.cd}`
Make ⟨*key*⟩ the default prefix.

1.4 Mathematical Expressions

See the Ti*k*Z/PGF manual for a detailed description.
Use parenthesis, `(` and `)`, for grouping. Arguments and values of trigonometric functions are in degree angle.
Arithmetic Operators: `+`, `-` (also unary minus), `*`, `/`, `^` (exponentiation), `!` (factorial, postfix operator), `r` (radian, postfix operator, see `deg`).
Relational Operators: `==`, `!=`, `<`, `<=`, `>`, `>=`.
Logical Operators: `!` (not, prefix operator), `||` (or), `&&` (and).
Conditionals: ⟨*condition*⟩`?`⟨*true*⟩`:`⟨*false*⟩.
Constants: `pi`, `e`, `false`, `true`.
Unary Functions: `abs`, `sign`, `int`, `frac` (fractional part), `round`, `floor`, `ceil`, `factorial` (see `!`), `iseven`, `isodd`, `isprime`, `sqrt`, `exp`, `ln`, `log10`, `log2`, `sin`, `cos`, `tan`, `cot`, `sec`, `cosec`, `asin`, `acos`, `atan`, `deg` (degree from radian), `rad` (radian from degree), `sinh`, `cosh`, `tanh`.
Binary Functions: `div` (integer division), `mod`, `Mod` (unsigned result), `gcd`, `pow` (see `^`), `atan2`, `veclen` (vector length in $\mathbb{R}^2$).
n-ary Functions: `min`, `max`.
Pseudo-Random Number Functions (Uniform Distribution): `rnd` ($[0,1] \cap \mathbb{R}$), `rand` ($[-1,1] \cap \mathbb{R}$), `random(`*n*`)` ($[1,n] \cap \mathbb{N}$), `random(`*m*`,`*n*`)` ($[m,n] \cap \mathbb{Z}$).

2 Axis Environments

`\begin{axis}[`⟨*axis options*⟩`]`?
⟨*axis options*⟩ → ⟨*key/value list*⟩

$\texttt{axis}_{\text{env}}$ can also be $\texttt{semilogxaxis}_{\text{env}}$, $\texttt{semilogyaxis}_{\text{env}}$, or $\texttt{loglogaxis}_{\text{env}}$.

▷ `every` ⟨*type*⟩? `axis` **style**
⟨*type*⟩ → (`linear`|`semilogx`|`semilogy`|`loglog`)
Define default axis options.

▷ `xmode|ymode|zmode = normal|linear|log` **option**
Customize axis scaling; `linear` is a synonym for `normal`.

▷ `log basis (x|y|z) =` ⟨*empty*⟩|⟨*positive number*⟩ **option**
The basis for logarithmic axis scaling. Empty means to apply the natural logarithm (base *e*) to any input coordinate – if the axis scaling is logarithmic – and use the decadic/common logarithm (base 10) for displaying tick labels. Any non-empty value causes both, coordinates and tick labels, to use the logarithm with base ⟨*number*⟩.

3 Plots

`\addplot[`⟨*plot options*⟩`]`? ⟨*input data*⟩ ⟨*trailing TikZ path commands*⟩`;`

`\addplot` (without options) and `\addplot+[`⟨*plot options*⟩`]` utilize default options from the cycle list. `\addplot[`⟨*plot options*⟩`]` only use the manually provided options.

▷ `every axis plot (no` *n*`)`? **style**
Define ⟨*plot options*⟩ for all plots or for the n^{th} plot of every axis. Plot numbers are zero-based.

3.1 Input Data

▷ `empty line = auto|none|scanline|jump` **option**
How to handle empty lines in ⟨*coordinates list*⟩, `none` means to do nothing, `jump` means to insert a discontinuity.

3.1.1 Coordinates List

⟨*input data*⟩ → `coordinates {`⟨*coordinates list*⟩`}`
⟨*coordinates list*⟩ → ⟨*coordinates*⟩*
⟨*coordinates*⟩ → `(`*x*`,` *y*`,` *z*`)` `(+- (`*u*`,` *v*`,` *w*`))`? `([`⟨*meta data*⟩`])`?

Read input data from a sequence of coordinates. *x*, *y*, and *z* are the point coordinates. *u*, *v*, and *w* are the error coordinates (reliability bounds) for error bar plots. Coordinate *z* and *w* are only mandatory for 3D plots. Empty lines in the ⟨*coordinates list*⟩ indicate discontinuities; use `\\` when gathering coordinates in a TEX macro.

▷ `plot coordinates/math parser = true|false` **option**
Whether or not to enable mathematical expressions in every coordinate inside of a ⟨*coordinates list*⟩.

3.1.2 Table Data

⟨*input data*⟩ → `table [`⟨*table options*⟩`]`? `{`⟨*table data*⟩`}`
⟨*table data*⟩ → ⟨*file name*⟩|⟨*inline table*⟩

Read input data from table columns.

▷ `table/`⟨*coordinate*⟩ `=` ⟨*column name*⟩ **option**
▷ `table/`⟨*coordinate*⟩ `index =` ⟨*column index*⟩ **option**
▷ `table/`⟨*coordinate*⟩ `expr =` ⟨*expression*⟩ **option**
⟨*coordinate*⟩ → `x|y|z|(x|y|z) error (plus|minus)`?`|meta`

Column names are case sensitive and have to exist. Use `{`⟨*column name*⟩`}` to quote non-trivial column names. The first column has index zero. Within ⟨*expression*⟩ `\thisrow{`⟨*column name*⟩`}` and `\thisrowno`⟨*column index*⟩ yields the cell value of the specified column. Likewise, `\coordindex` yields the index of the current set of coordinates and `\lineno` yields the total line number. Both numbers start counting at zero.

▷ `table/header = true|false` **option**
Whether or not to check ⟨*table data*⟩ for column names. If enabled, the first non-comment line is checked for column names. That means if any element is not a number, all entries are treated as column names.

▷ `table/skip first n = 0|`⟨*non-negative integer*⟩ **option**
Don't process the first *n* lines in ⟨*table data*⟩.

▷ `table/ignore chars = {}|`⟨*comma-separated list*⟩ **option**
▷ `table/white space chars = {}|`⟨*comma-separated list*⟩ **option**
▷ `table/comment chars = {}|`⟨*comma-separated list*⟩ **option**
Extra characters to be ignored, treated like a whitespace character (beside space and tab), or treated like a comment start character (beside `#` and `%`).

▷ `table/row sep =` ⟨*newline*⟩`|\\` **option**
Use `\\` as the row seperator if you experience problems with ⟨*newline*⟩, for example with inline table data or when gathering table data in a TEX macro.

▷ `table/col sep = space|tab|comma|semicolon|colon`↵ **option**
`|braces|&|ampersand`

A `space` column separator means one or more space or tab characters. With `braces`, every table cell looks like `{`⟨*contents*⟩`}` and whitespace characters between adjacent table cells is ignored. A `&` column separator implies '`table/trim cells = true`'.

▷ `table/read completely = auto|true|false` **option**

Whether or not to read the whole table into memory. Use with care!

▷ `table/search path = {}|`⟨*comma-separated list*⟩ **option**
▷ `table/search path/implicit . = true|false` **option**

Search path for input files, `.` means to use the standard TEX procedure.

`\pgfplotstableread{`⟨*file name*⟩`}\foo`
`\addplot table [`⟨*table options*⟩`] {\foo};`

Read table data once so that you can use it multiple times; `\foo` is a user-defined command sequence.

3.1.3 Mathematical Expressions

⟨*input data*⟩ → `expression`$^?$ `{`⟨*expression*⟩`}`
⟨*input data*⟩ → `(`⟨*x-expression*⟩`, `⟨*y-expression*⟩`, `⟨*z-expression*⟩`)`

Create input data by sampling a mathematical expression over an argument domain. The second form can be used to create parametric plots. Say `{`⟨*x-expression*⟩`}` if ⟨*x-expression*⟩ contains parenthesis or commas. The ⟨*z-expression*⟩ is only mandatory for 3D plots.

▷ `domain = -5:5|`⟨x_1⟩`:`⟨x_2⟩ **option**
▷ `domain y = `⟨*empty*⟩`|`⟨y_1⟩`:`⟨y_2⟩ **option**

Define the argument domain for the x-axis to the closed interval $[x_1, x_2]$. Likewise for the y-axis for 3D plots. If `domain y` is empty, use the value of `domain`.

▷ `samples = 25|`⟨*non-negative integer*⟩ **option**
▷ `samples y = `⟨*empty*⟩`|`⟨*non-negative integer*⟩ **option**

The number of samples to be generated. Samples are equally spaced over the corresponding argument domain. If '`samples y`' is empty, use the value of `samples`.

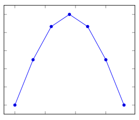
▷ `samples at = {}|`⟨*comma-separated list of numbers*⟩ **option**

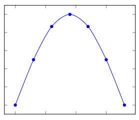
Explicit argument values for sampling ⟨*expression*⟩. This option always overrides the `domain` and `samples` options.
⟨*comma-separated list of numbers*⟩ can contain `...` expressions, for example '`{-2, -1.8, ..., 2}`'.

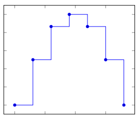
▷ `variable = x|`⟨*variable name*⟩ **option**
▷ `variable y = y|`⟨*variable name*⟩ **option**

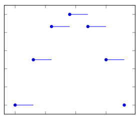
The variable name containing the argument value when evaluating ⟨*expression*⟩.

3.2 Line Plots


sharp plot


smooth


const plot


jump mark left

▷ `/tikz/sharp plot` **option**

Connect points by straight lines. This is the default.

▷ `/tikz/smooth` **option**
▷ `/tikz/tension = 0.55|`⟨*number*⟩ **option**

Connect points by a smooth curve. For best results, points should be equidistant and the bending angles should be less than about 30°. The `tension` option controls the sharpness of the corners; 0 yields sharp corners and 1 yields a circle if the path is a square.

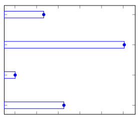
▷ `/tikz/const plot` **option**
▷ `/tikz/const plot mark (left|mid|right)` **option**

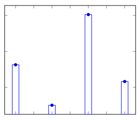
Connect points with horizontal and vertical line segments. '`const plot`' is an alias for '`const plot mark left`'. Markers are placed on the left corner, in the middle, or on the right corner of the horizontal line segments. Use '`const plot, no markers`' to omit the markers.

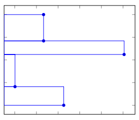
▷ `/tikz/jump mark (left|mid|right)` **option**

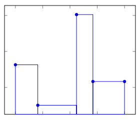
Like '`const plot`' but omit the vertical line segments.

3.3 Bar Plots


xbar


ybar


xbar interval


ybar interval

▷ `/tikz/xbar` **option**
▷ `/tikz/ybar` **option**

Render coordinates as horizontal or vertical bars respectively.

▷ `/pgf/bar width = 10pt|`⟨*dimension*⟩`|`⟨*number*⟩ **option**

Width of a single bar. ⟨*dimension*⟩ is a TEX dimension and ⟨*number*⟩ is in axis units. Value can be a mathematical expression. The fully computed value is then available in `\pgfplotbarwidth`.

▷ `/pgf/bar shift = 0pt|`⟨*dimension*⟩`|`⟨*number*⟩ **option**

Off-center distance for the bars. ⟨*dimension*⟩ is a TEX dimension and ⟨*number*⟩ is in axis units. Value can be a mathematical expression. The fully computed value is then available in `\pgfplotbarshift`.

▷ `xbar` **style**
▷ `xbar( = 2pt|`⟨*dimension*⟩`|`⟨*number*⟩`)`$^?$ **option**
▷ `ybar` **style**
▷ `ybar( = 2pt|`⟨*dimension*⟩`|`⟨*number*⟩`)`$^?$ **option**

Predefined axis style for bar plots; implies `/tikz/xbar` or `/tikz/ybar` respectively, `bar shift auto`$_{\text{sty}}$, and `bar cycle list`$_{\text{sty}}$. The default handler takes one optional argument which is passed on to `bar shift auto`$_{\text{sty}}$.

▷ `bar shift auto` **style**
▷ `bar shift auto = 2pt|`⟨*dimension*⟩`|`⟨*number*⟩ **option**

Predefined axis style setting /pgf/bar shift to the correct value based on the current plot number and the total number of plots. Argument is the distance between adjacent bars of a group.
When n bar plots are added to an axis, the total width for a group of bars is $n \times$ ⟨*bar width*⟩ $+ (n-1) \times$ ⟨*bar shift auto*⟩.

▷ `bar cycle list` **style**

Predefined axis style installing a cycle list for bar plots.

▷ `bar direction = auto|x|y` **option**

Explicitly set the bar plot direction. Not needed if you say, for example '`ybar, bar width = 1`', because the direction is clear from the context.

▷ `/tikz/xbar interval` **option**
▷ `/tikz/ybar interval` **option**

Like /tikz/xbar or /tikz/ybar respectively, but draw the bar width as an interval from this point to the next point. You need one extra point to define the interval for the last bar.

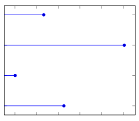
▷ `xbar interval` **style**
▷ `xbar interval( = 1|`⟨*relative width*⟩`)`$^?$ **option**
▷ `ybar interval` **style**
▷ `ybar interval( = 1|`⟨*relative width*⟩`)`$^?$ **option**

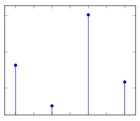
Predefined axis style for interval bar plots; implies `/tikz/xbar interval` or `/tikz/ybar interval` respectively and `bar cycle list`$_{\text{sty}}$. The default handler takes one optional argument to scale the intervals.

▷ `xticklabel interval boundaries` **style**
▷ `yticklabel interval boundaries` **style**
▷ `zticklabel interval boundaries` **style**

Axis style to display the interval bounds in the tick labels.

3.4 Comb Plots


xcomb


ycomb

▷ `/tikz/xcomb` **option**
▷ `/tikz/ycomb` **option**

Render coordinates as horizontal or vertical lines respectively.

3.5 Quiver Plots

▷ `quiver = {`⟨*quiver options*⟩`}` **option**

Render coordinates as small arrows. The origin of the arrow is at the final point coordinates (x, y, z) and the direction and length of the arrow is defined by the direction coordinates (u, v, w).
The `quiver/` prefix can be omitted within ⟨*quiver options*⟩.

▷ `quiver/(u|v|w) = `<u>`0`</u>`|`⟨*expression*⟩ **option**

The direction coordinates of the arrows. Within ⟨*expression*⟩, `x`, `y`, and `z` are bound to the final point coordinates.
For parametric plots use '`variable = t`' and '`quiver/u = f(t)`' and '`quiver/v = g(t)`' to access the parameter.

```
\addplot[
  variable = t,
  quiver = {u = {-sin(t)}, v = {cos(t)}},
]
({cos(t)}, {sin(t)});
```

▷ `quiver/(u|v|w) value = `<u>`0`</u>`|`⟨*number*⟩ **option**

Like `quiver/u`, `quiver/v`, and `quiver/w` respectively but without parsing mathematical expressions. However, `\thisrow{`⟨*column name*⟩`}` and similar code works.

▷ `quiver/colored` **option**
▷ `quiver/colored = `<u>`mapped color`</u>`|`⟨*color*⟩ **option**

Set a different color for each arrow. `quiver/colored` is an alias for '`quiver/colored = mapped color`'. Please note that '⟨*color*⟩`, quiver = `. . .' is more efficient if ⟨*color*⟩ is constant.

▷ `quiver/scale arrows = `<u>`1`</u>`|`⟨*number*⟩ **option**

Scale all arrows by a constant factor.

▷ `quiver/update limits = `<u>`true`</u>`|false` **option**

Whether or not the coordinates of the arrow heads shall be considered when determining the axis limits.

▷ `quiver/every arrow` **style**

Style to customize arrows individually at visualization time.

▷ `quiver/before arrow` **code**
▷ `quiver/after arrow` **code**

Run ⟨*TEX code*⟩ before and after drawing a single arrow. Empty by default.

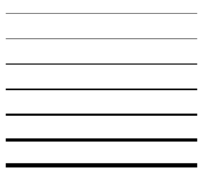
▷ `quiver/quiver legend` **style**

Style that redefines `legend image code` in order to produce a suitable legend for quiver plots.

4 Lines and Markers

4.1 Line Width

▷ `/tikz/ultra thin` **style**
▷ `/tikz/very thin` **style**
▷ `/tikz/thin` **style**
▷ `/tikz/semithick` **style**
▷ `/tikz/thick` **style**
▷ `/tikz/very thick` **style**
▷ `/tikz/ultra thick` **style**



Predefined line widths.

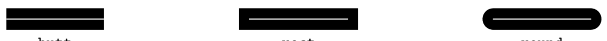
▷ `/tikz/line width = `<u>`0.4pt`</u>`|`⟨*dimension*⟩ **option**

Set the line width.

4.2 Line Cap

▷ `/tikz/line cap = `<u>`butt`</u>`|rect|round` **option**

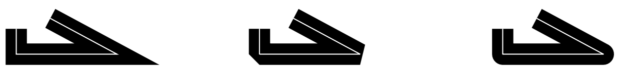
Set the line cap style.



4.3 Line Join

▷ `/tikz/line join = `<u>`miter`</u>`|bevel|round` **option**

Set the line join style.

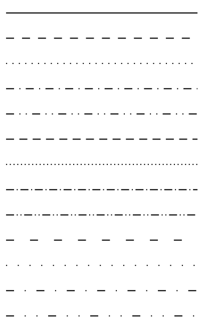


▷ `/tikz/miter limit = `<u>`10`</u>`|`⟨*number*⟩ **option**

When the ratio of the miter length to the line width is greater than ⟨*number*⟩, the miter join is replaced by a bevel. A miter limit $\ell = 1/\sin(\alpha/2)$ for $\alpha \in (0°, 180°]$ will create a bevel join for angles less than $\alpha = 2 \cdot \arcsin(1/\ell)$.

4.4 Dash Pattern

▷ `/tikz/solid` **style**
▷ `/tikz/dashed` **style**
▷ `/tikz/dotted` **style**
▷ `/tikz/dashdotted` **style**
▷ `/tikz/dashdotdotted` **style**
▷ `/tikz/densely dashed` **style**
▷ `/tikz/densely dotted` **style**
▷ `/tikz/densely dashdotted` **style**
▷ `/tikz/densely dashdotdotted` **style**
▷ `/tikz/loosely dashed` **style**
▷ `/tikz/loosely dotted` **style**
▷ `/tikz/loosely dashdotted` **style**
▷ `/tikz/loosely dashdotdotted` **style**



Predefined line styles.

▷ `/tikz/dash pattern = `((`on|off`) ⟨*dimension*⟩)$^+$ **option**

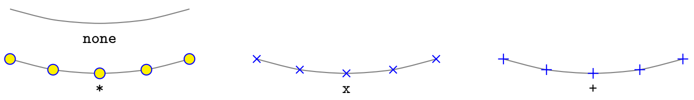
Set the dash pattern (line style) for drawing lines, e.g., '`dash pattern = on 3.5mm off 0.7mm`'.

▷ `/tikz/dash phase = `<u>`0pt`</u>`|`⟨*dimension*⟩ **option**

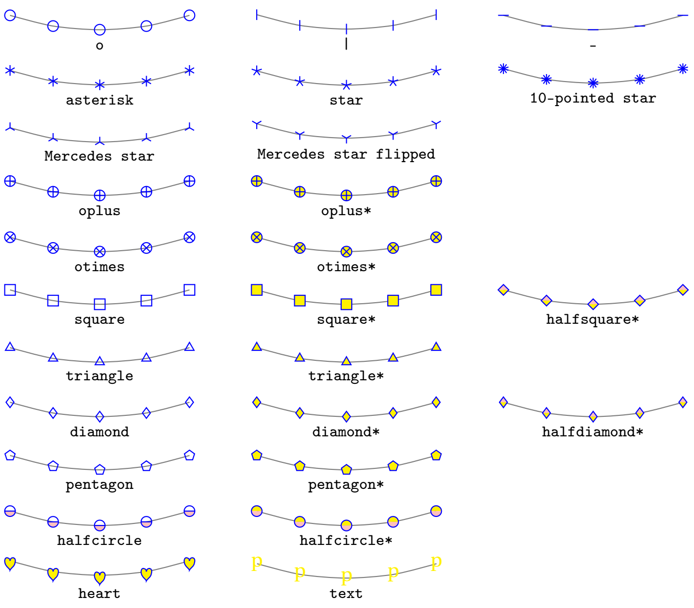
Start the dash pattern at offset ⟨*dimension*⟩.

4.5 Markers

Standard markers:



With `\usetikzlibrary{plotmarks}`:



All markers plotted with '`mark options = {draw = blue, fill = yellow}`' and '`mark color = pink`'. You can rotate makers with, e.g., '`mark options = {rotate = 90}`'.

▷ `/tikz/mark = `<u>`*`</u>`|`⟨*marker*⟩ **option**

Use ⟨*marker*⟩.

▷ `/tikz/mark size = `<u>`2pt`</u>`|`⟨*dimension*⟩ **option**

Marker size, ⟨*dimension*⟩ is either the radius or about half the width or height.

▷ `/tikz/mark repeat = `<u>`1`</u>`|`⟨*integer*⟩ **option**

Draw a marker at every ⟨*integer*⟩th sample.

▷ `/tikz/mark phase = `<u>`1`</u>`|`⟨*integer*⟩ **option**

Draw the first marker at the ⟨*integer*⟩th sample; ⟨*integer*⟩ is one based.

▷ `/tikz/mark indices = `<u>`{}`</u>`|{`⟨*comma-separated list of integers*⟩`}` **option**

Explicit sample indices for drawing markers. ⟨*comma-separated list of integers*⟩ can contain `...` expressions, for example '`mark indices = {1, 2, ..., 7}`'.

▷ `/tikz/every mark` **style**

This style is applied before drawing a marker.

▷ /tikz/`mark options = {`⟨*options*⟩`}` **option**

Redefine ‘`every mark`’ so that it sets ⟨*options*⟩.

▷ /pgfplots/`no markers` **style**

Disable markers; even for cycle lists that contain markers.

▷ /pgf/`mark color = `<u>`white`</u>|⟨*color*⟩ **option**

Additional fill color for `halfcircle`, `halfcircle*`, `halfdiamond*`, and `halfsquare*` markers.

▷ /pgf/`text mark = `<u>`p`</u>|⟨*text*⟩ **option**

Define the text for ‘`mark = text`’.

▷ /pgf/`text mark as node = `<u>`false`</u>|`true` **option**

Whether or not to draw text markers as nodes.

▷ /pgf/`text mark style = {`⟨*options*⟩`}` **option**

Customize the appearance of text markers. When ‘`text mark as node`’ is true, ‘`text mark style`’ are `\node` options. Otherwise, ‘`text mark style`’ are `\pgftext` options.

4.6 Colors

Color support is provided by the `xcolor` package. Standard color names:

⬛ `black`	🟥 `red`	🟩 `green`	🟦 `blue`
⬛ `darkgray`	🟦 `cyan`	🟪 `magenta`	🟨 `yellow`
⬛ `gray`	🟫 `brown`	🟩 `lime`	🟫 `olive`
⬜ `lightgray`	🟧 `orange`	🟥 `pink`	🟥 `purple`
⬜ `white`	🟩 `teal`	🟪 `violet`	`none`

▷ /tikz/`color = `⟨*color*⟩ **option**

Set the color for drawing and filling. You can omit the option key if ⟨*color*⟩ is a color name.

▷ /tikz/`draw = `⟨*color*⟩ **option**
▷ /tikz/`fill = `⟨*color*⟩ **option**

Set the color for drawing or filling respectively. You can use `none` as ⟨*color*⟩ to disable drawing or filling.

`\definecolor{`⟨*name*⟩`}{`⟨*model*⟩`}{`⟨*spec*⟩`}`
⟨*model*⟩ → `rgb`|`cmy`|`cmyk`|`hsb`|`Hsb`|`tHsb`|`gray`|`RGB`|`HSB`|`Gray`|`HTML` ↲
    |`wave`
⟨`rgb` *spec*⟩ → x`,` x`,` x
⟨`cmy` *spec*⟩ → x`,` x`,` x
⟨`cmyk` *spec*⟩ → x`,` x`,` x`,` x
⟨`hsb` *spec*⟩ → x`,` x`,` x
⟨`Hsb` *spec*⟩ → H`,` x`,` x
⟨`tHsb` *spec*⟩ → H`,` x`,` x
⟨`gray` *spec*⟩ → x
⟨`RGB` *spec*⟩ → L`,` L`,` L
⟨`HSB` *spec*⟩ → M`,` M`,` M
⟨`Gray` *spec*⟩ → N
⟨`HTML` *spec*⟩ → $[000000_{16}, \mathrm{FFFFFF}_{16}]$
⟨`wave` *spec*⟩ → $[363, 814]$

$x = [0, 1]$, $H = [0, 360]$, $L = [0, 255] \cap \mathbb{Z}$, $M = [0, 240] \cap \mathbb{Z}$, and $N = [0, 15] \cap \mathbb{Z}$. All colors are defined in the sRGB color space. HSB is a synonym for HSL.

Option Index

Concept Index