

PGFPlots

Cheatsheet, page 1 of 2

1 Pgfplots

Syntax

```
\begin{tikzpicture}
\begin{axis}[<ax_option>*]
[\addplot[+|3|]<p1_option>*<plot>]+
[\draw{...}]*
[\node{...}]*
[\legend[]]?
\end{tikzpicture}
```

There are a large number of `<ax_option>` and `<p1_option>`s, each of which are of the following types: *flags*^F, *styles*^S, or *keys*^K. All options can be set for an axis or plot as indicated, and many can be set for different scopes (optionally delimited with {} braces) using `\pgfplotsset{...}`. Assign `\empty` for non-drawn entity. Each option is described below, in the sections relevant to the graphical behavior controlled.

Examples

Many more examples [here](#)!

```
\begin{tikzpicture}           % tikz envmt
\begin{axis}                 % pgfplot axis
[legend cell align=left,     % <ax_opts>
legend pos=outer north east] % ...
\addplot coord's {(0,0) (1,1)}; % coords
\addplot+[...] coord's ...;    % appnd opts
\addplot {sin(x)};             % math expr
\legend{a,fine,legend}        % legend
\end{axis}                   %
\end{tikzpicture}            %
```

Plot-level Settings

[ultra]thick <color>

2 Addplot Command

Syntax

```
\addplot coordinates (<tuple>)*
\addplot fill between[... ]
\addplot table {file.dat}
\addplot table <inline_data_row>*
\addplot {<math_expression>}
\addplot gnuplot {<gnuplot_code>}
\addplot graphics {<image_file>}
```

Coordinates

Tables

Map dataset columns to plot variables using **x**, **y** keys. Further processing available with **x expr**, **y expr** keys. Inside such expressions, the following helper macros are available: `\thisrow` `\thisrowno` `\coordindex` `\lineno`

Expressions

By default, will plot “marks” at 20 “samples” along the function described by `<expression>`,

connecting each with piecewise line segments. Use `smooth` for Bezier interconnections, instead.

```
\addplot[smooth,samples=50,mark=none]
{x^2};
```

Δ default to radians using `trig` format plots.

Gnuplot

```
\addplot gnuplot [id=exp,domain=0:10]
{exp(x)};
\addplot3[contour gnuplot][exp(0-x^2-y^2)];
```

Options exclusive to gnuplotting in pgfplots:

<code>translate gnuplot</code>	<code>parametric</code>
<code>id</code>	<code>parametric/var 1d</code>
<code>prefix</code>	<code>parametric/var 2d</code>
<code>raw gnuplot</code>	

Fill Between

Do something like:

```
\addplot[gray] fill between
[of=A and B,soft clip=(domain=3:4),];
... where A and B are previously-constructed tikz
or pgfplots “named paths.”
```

Graphics Plotting

Position image with `[x|y][min|max]`; process with `\includegraphics`, which passes parameters to `\node[]{\includegraphics[...]{...}}`. Control default node parameters with `node/.style key`.

```
\addplot graphics
[xmin=0,xmax=1,ymin=0,ymax=1,
includegraphics={trim=12 9 12 8,clip}]
{external.file};
```

It is possible to overlay PGFPlots axes onto 3D graphics, but is more tedious, and requires mapping 3D points to 2D canvas points. See [here](#).

Parametric Plots

Can change `variable`, but must set `domain`:

```
\addplot [variable=t,domain=0:2*pi]
{(sin(t),2*cos(t))}
```

3 Axes & Legends

All of the below can be set in styles rather than set on each plot or axis. ∃ a number of pre-fab styles for axes, lines, ticks, legends, and colorbars, such as: **every major tick**, **title style**, **every loglog axis**, etc. See [here](#) for a full list. Or, ideally, create your own style, appending it to the relevant elements as call for using: `\pgfplots-set{<style_name>/.style=...}`.

Axis Lines & Labels

The following options afford setting axis and plot titles, including their style and positioning.

<code>title</code>	<code>hide [x y z] axis</code>
<code>extra description</code>	<code>inner axis line style</code>
<code>axis lines</code>	<code>outer axis line style</code>
<code>axis lines*</code>	<code>grid</code>
<code>axis [x y z] line</code>	<code>domain</code>
<code>axis [x y z] line*</code>	<code>[x y z][min max]</code>
<code>every inner [x y z] line</code>	<code>[x y z]label</code>
<code>every outer [x y z] line</code>	<code>[x y z]label shift</code>
<code>axis line style</code>	<code>[x y z]label near ticks</code>
<code>[x y z] axis line style</code>	<code>[x y z]label absolute</code>
<code>every boxed [x y z] axis</code>	<code>axis line style</code>
<code>separate axis lines</code>	<code>legend pos</code>
<code>axis [x y z] line shift</code>	<code>enlargelimits</code>
<code>axis [x y z] discont'y</code>	

Tick Marks

<code>[x y z]tick</code>	<code>minor [x y z] tick num</code>
<code>[x y z]ticklabel</code>	<code>extra [x y z] ticks</code>
<code>[x y z]ticklabels</code>	<code>[x y z]tickten</code>
<code>"" from table</code>	<code>scaled [x y z] ticks</code>
<code>extra [x y z]tick label</code>	<code>max space between ticks</code>
<code>[x y z]minorticks</code>	<code>min space between ticks</code>
<code>[x y z]majorticks</code>	<code>try min ticks</code>
<code>[x y z]tickmin</code>	<code>tickwidth</code>
<code>[x y z]tickmax</code>	<code>major tick length</code>
<code>[x y z]tick pos</code>	<code>minor tick length</code>
<code>[x y z]ticklabel pos</code>	<code>subtickwidth</code>
<code>[x y z]tick align</code>	<code>[x y z]tick placement to-lerance</code>
<code>[x y z]tick distance</code>	<code>hide obscured [x y z] ticks</code>
<code>[x y z]ticklabel shift</code>	<code>sloped like [x y z] axis</code>
<code>[x y z]ticklabel style</code>	<code>every x tick scale label</code>
<code>minor [x y z] tick⁵</code>	

3D Axes

Radial Axes

Legends

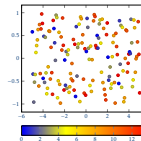
Create imperatively using `\addlegendentry` or `\legend`, or declaratively with option `legend entries`. Behind scenes, legends are implemented as Tikz matrices, so the styling for those apply here as well. Customize placement & appearance:

<code>at</code>	<code>[x y z]bar legend</code>
<code>anchor</code>	<code>[x y z]bar interval le-gend</code>
<code>legend pos</code>	<code>legend plot pos</code>
<code>mesh legend</code>	<code>legend cell align</code>
<code>reverse legend</code>	<code>legend columns</code>
	<code>transpose legend</code>

Colorbars

These can be positioned, given different color scales, [dis]connected to plotting variables, moved outside an individual plot, etc. Often associated with `point meta` data.

<code>colorbar right</code>	<code>colorbar shift</code>
<code>colorbar top</code>	<code>colorbar style</code>
<code>colorbar horizontal</code>	<code>colorbar/width</code>
<code>every colorbar</code>	<code>colorbar source</code>
<code>point meta [min max]</code>	<code>colorbar sampled</code>
<code>colorbar to name</code>	<code>colorbar as legend</code>



Axis Coordinate Systems

∃ two coordinate systems, unique to PGFPlots that allow node positioning along axis locations, namely, **axis description cs** and **[x|y|z]ticklabel cs**. The former is simple, the latter affords better placement along 3D axes. Additionally, these coordinate systems enable some extra anchors for nodes: `near [x|y|z]ticklabel`, `near ticklabel` opposite.

4 Positioning in Document

Scaling

To size and scale a plot, use either `width` and `height`, or position along unit vectors as indicated with **x**, **y**, and **z**. Related general scaling, sizing, & aspect-ratio options include:

<code>width</code>	<code>[x y z] dir</code>	<code>axis equal image</code>
<code>height</code>	<code>scale only axis</code>	<code>unit vector ratio</code>
<code>x</code>	<code>scale mode</code>	<code>unit vector ratio*</code>
<code>y</code>	<code>z mode</code>	<code>[x y z] post scale</code>
<code>z</code>	<code>axis equal</code>	

Some pre-fab scaling styles for controlling various axis & font settings: `normalsize`, `small`, `footnotesize`, and `tiny`.

Additionally, low-level control is afforded through several scaling “strategies”.

Alignment

Bounding Box

Layers

Grouping

5 Styling

Colors

Marks

line patterns

Number Formatting

Annotations & Labeling

`symbolic y coords`
`nodes near coords`

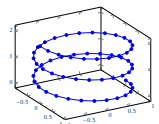
Tikz Interaction

Use disabled `datascaling opt'n` to coerce tikz to pgfplots coord system. Then Tikz accordingly, eg: `\draw[thick,<->] ($(\x,{sin(2 * \x)})$) -- ...`

6 Plot Types

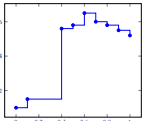
Line

Direct piecewise connections between samples are the default. Use `sharp plot` to indicate explicitly, or change to `smooth` for Bezier interconnections. Create 3D line plots by passing 3d tuples into `\addplot3`.



Constant

Graphs with horizontal line connections. Options control placement of segments relative to coordinates, and whether vertical segments also shown.



- const plot mark left

const plot mark right

const plot mark mid
- jump mark left

jump mark right

jump mark mid

Bar

`[x|y]bar`, indicated at `axis` level, affords bar charts in either `x` or `y` direction. Each `\addplot` contributes an additional series. Options apply at axis level as well (except those marked with *):

`xbar` bar width
`ybar` enlarge `[x|y]` limits
`bar shift auto` pattern
`update limits` bar cycle list

Use `bar cycle list` to install new styles for different series. A different graph altogether is `[x|y]bar interval` which, together with `[x|y|z]ticklabel interval boundaries` allows for bars of differing widths.

Histogram

Hist's take `\addplot table` or `<expression>` commands. Data must be single column.

`\addplot+ [hist={bins=3}]`
`table ...`

`\addplot+ [samples=200,hist] {rnd};`

`data` intervals
`data min` cumulative
`data max` density
`bins` handler
`symbolic coords` data filter
`data coord trafo`

Box

Can adjust orientation, box thickness, outlier distance, quartile thresholds, "whisker" styling, etc, with the following:

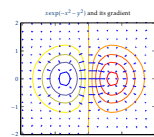
Comb

Similar to bar chart, but with lines rather than bars. As with bar, indicate with a set of coordinates:

`\addplot+[xcomb] coordinates ...`

Quiver

Requires a vector input (u, v) that will calculate respective (x, y) components of each "quiver. Works readily with parametrics (using `variable=t`) & tables of predefined (x, y, u, v) vectors.



`u` colored
`v` scale arrows
`w` update limits
before arrow
after arrow
quiver legend
every arrow

Often used to visualize tangent & gradient fields (using 3D quivers). Set `w` for 3rd dimension.

Stacked

Initiate basic stacking with `stack plots=y|x`, or stacked bars with, eg, `ybar stacked`. Given coordinates for each `\addplot` line up positionally to comprise each of the plot's "series'." Can also stack in negative direction.

`ybar stacked` reverse stacked plots
`xbar stacked` stacked ignores zero
`stack dir` xbar interval stacked
`stack negative` ybar interval stacked
`bar shift auto`

Mesh

Available as 2D or, as is more customary, 3D. Invoke with `mesh`. Default mapping from `y` to `colormap`. Change this using `point meta`.

Area

For simple areas use stacking in addition to `area style`. Can also use `\addplot fill` between in other plot types to achieve a fill affect, or use `\closedcycle` at end of `\addplot` to close a drawn path. Further restrict areas by setting `soft clip` key to a previously-defined "clip."

Scatter

For simple scatter plots, use `only marks`. For more control over marks, including styling (eg by color), and "classing" into groups, use `scatter`. "Classes" allow styling by ordinal metadata set using `scatter src` or `point meta`. Easily extended to 3D axes, which consume 3d vectors.

`scatter src` use mapped color
`classes` scatter position

Ternary Diagram & Tieline

Requires "ternaryaxis" package. Plots to a "Barycentric" coordinate system, requiring that relative coordinates that are correctly specified.

"Tielines" are binodal curves in a ternary coordinate plane.

every ternary axis ternary limits relative
`tieline` tieline style
`curve style`

Smith

Requires external "smithchart" library. Can plot coords or lines onto this coordinate system, can invert the chart into a "admittance" chart, and can manipulate size and ticks. Low-level grid-line control with `x|ygrid` each `n`th passes `y` and `x|ygrid stop` at `x|y`.

smithchart mirrored
show origin
ytick label ar'nd circle
ytick label in circle
many smithchart ticks
few smithchart ticks
every smithchart axis
default s.c. x|y tick

Contour

3D contours, like topo charts. Use `contour prepared` or `contour gnuplot` for contour input, the former which takes matrix-style input, and the latter which creates these matrices and passes them to the former.

`number` contour prep'd format
`levels` contour dir
`draw color` contour label style
`labels` label distance
`contour external` labels over line
`contour gnuplot` label node code
`contour filled`

Surface

Visualize 3D surfaces, which are—unlike with meshes—filled. Easiest to create with `<expression>` but can also create with table or coordinates. Choosing from various "shaders" affords control over color gradients on each patch. Colormaps are inherited from the axis, but can be overridden. For lower-level control of color interpolations, set `mesh/color` input. This works with `mesh`, `patch` or `surf`.

Patch

Regression