

The tikzfxgraph Package

simplified $f_i(x)$ graphics

Version 1.1

Alceu Frigeri*

November 2025

Abstract

This package offers a set of streamlined commands to draw algebraic functions, atop of *pgfplots* and *gnuplot*. Some auxiliary commands are also defined allowing to create *sets of functions* and *user defined styles*.

Contents

1	Introduction	1
2	Requirements	1
3	Package Options	2
4	Commands	2
4.1	Defining Functions' sets	2
4.2	User Defined Styles	3
4.3	Drawing a $f_i(x)$ Graph	4
5	Style's Default	5
6	Examples	6
6.1	Drawing a Bode Diagram	6
6.2	A Few Curves at Once	7
6.3	Further Customization	8

1 Introduction

The *pgfplots*[1] package together with *gnuplot*[3] is extremely flexible, allowing the construction of very intricate graphics, but said flexibility comes at a cost: the sheer number of parameters to be set. This package is nothing more than a wrap around *pgfplots* and *gnuplot* offering a somewhat simplified interface.

A command (`\fxgraphdraw`) and an environment (`fxgraph`) are provided for drawing one or more $f_i(x)$ curves, as well some auxiliary commands (`\fxsetnew` and `\fxsetappend`) to define sets of functions and one for user defined styles (`\fxsetnewstyle`).

2 Requirements

One needs to load at least two packages: `tikz` and `pgfplots` (it is advisable to set the *pgfplots* compatibility to, at least, 1.18 (`\pgfplotsset{compat=1.18}`)). Like

LaTeX Code:

```
\usepackage{tikz}
\usepackage{pgfplots}
\pgfplotsset{compat=1.18}

\usetikzlibrary{pgfplots.units} %this is optional
```

*<https://github.com/alceu-frigeri/tikzfxgraph>

Besides that, *gnuplot* must be installed/present.

Note: to be able to call *gnuplot* you will need to use the `-enable-write18` (or `-shell-scape`) option in your L^AT_EX run. Keep in mind that *gnuplot* will create a set of files that can be used from one run to another, and, unless you change the domain/functions, `-enable-write18` (or `-shell-scape`) can be kept disable otherwise.

Warning: As of now, there is a bug on version 6.0.*, under windows, of *gnuplot*. It will work, but a series of errors messages will be raised (invalid characters) that's due one of the returning files being written in UTF16 instead of UTF8. So, for now, use at most version 5.4.* <https://sourceforge.net/p/gnuplot/bugs/2747/>

3 Package Options

- `samples default` (default: 2000) Globally sets the default number of samples to be calculated. This can be over-written per function definition (see 4.1).
- `no domain warning` (default: false) Given the many ways by which a function domain can be defined, in some cases this package will issue a warning if the domain wasn't explicitly defined (see 4.3), which might be (or not) an error. This option suppresses those warnings.
- `use file` (default: false) This will globally set `use file` for every `\fxgraphdraw` or `fxgraph` environment. The same conditions (per function) and warnings apply: The corresponding table files must exist but they won't be verified if they are up-to-date. With this global option set, one can selectively reverse it per `\fxgraphdraw` or `fxgraph` environment, by using `use file=false` with them.

If each and every function used has a corresponding *distinct id*, the option `use file` can be used to share the L^AT_EX 2_ε document with someone who doesn't have *gnuplot* installed (or doesn't want to enable `-enable-write18` or `-shell-scape`).

4 Commands

4.1 Defining Functions' sets

`\fxsetnew` `\fxsetnew {⟨new-fxset⟩}`

This defines/create a `⟨fxset⟩` for later reference. A `⟨fxset⟩` is just a "repository" of functions descriptions/specifications.

Note: About dataset's names: It can be almost anything, the name can contain strings normally not allowed in a macro name, like spaces, dots, two-dots and so on, including backslashes, meaning that if someone typesets `\XYZ` as a dataset, `\XYZ` will be it's name: a backslash isn't an active character anymore and one can't use macros when defining a `fxset`'s name.

`\fxsetappend` `\fxsetappend {⟨fxset⟩}{⟨keyval-list⟩}`

Adds a function description to a given `fxset`.

Valid Keys when describing a function:

- `fx` The function itself. It can be any *gnuplot* valid expression, in terms of *x*.
- `id` (optional) An unique identifier. A set of auxiliary files are created by *gnuplot* using this as a name suffix. As per the *pgfplot* manual, they are used to determined if there is the need to re-run *gnuplot*.
- `legend` (optional) The name of the function, to appears as a Legend.
- `samples` (default: 2000) The default value can be changed by the package option `samples default`) Set the number of points to be calculated.
- `use file` (optional) If `id` is defined, and the corresponding table file exists (`\jobname.<id>.table`), then the table file will be used instead of calling *gnuplot*. **Warning:** It won't check if the table file is correct/up to date! This is, mainly, a work around for some cases in which *pgf/pgfplot* fails to recognize that 'nothing has changed' and needless call *gnuplot*. This setting has precedence over the corresponding global/environment setting.

Note: An error is raised if `\fxset` isn't defined.

Note: Either set a legend for each and every function, or to none of them. Mixing styles (some with a legend, some without, will result in functions being wrongfully labelled).

Besides those, any other *pgfplots* valid key can be used. (e.g. `red` , `thick` : the specific curve will be in red, using a thick line). For example:

```
\fxsetnew{set-A} %creating a new 'set'

\fxsetappend{set-A}{
  id=f-A0 ,
  fx=x^2-x+2 ,
  thick %this key comes from tikz/pgf
}
\fxsetappend{set-A}{
  id=f-A1 ,
  fx=x^2+x+3 ,
  red %this key comes from tikz/pgf
}
```

4.2 User Defined Styles

`\fxsetnewstyle` `\fxsetnewstyle {<style-name>} {<keyval-list>}`

Defines a new *pgfplot* style key. Which can be later used when drawing $f_i(x)$ function graphs (can be used, for instance, to assure all graphs follow the same style).

Valid Keys when describing a function:

<code>linear</code>	Both x and y axis are linear.
<code>loglog</code>	Both x and y axis are logarithmic.
<code>semilog x</code>	The x axis is logarithmic, the y is linear.
<code>semilog y</code>	The y axis is logarithmic, the x is linear.
<code>x ticks</code>	Describes the minor ticks to be drawn in the x axis. See below.
<code>y ticks</code>	Describes the minor ticks to be drawn in the y axis. See below.

Besides those, any other *pgfplots* valid key can be used. (e.g. `red` , `thick` : the specific style will set the lines to be red and thick).

Note that the keys `x ticks` and `y ticks` are themselves defined by a set of keyval values, as follow:

<code>min</code>	The minimal value (starting value) of the corresponding axis.
<code>max</code>	It's maximum value.
<code>delta</code>	(optional) the <i>delta</i> to be used between <i>ticks</i> . Note that, it depends on the kind of the axis. In case of a linear axis, this is just the delta between ticks. If logarithmic, it is the geometric distance between ticks.
<code>N</code>	(optional) Sets the number of ticks to be calculated. In case of a linear axis it will set the linear distance to $(max - min)/N$. In case of a logarithmic axis it will set the geometric distance to $(\ln(max) - \ln(min))/N$. <i>N</i> has precedence over <i>delta</i> .
<code>units</code>	(optional) The units of the corresponding axis.

The following example will define a style “my style A”, to be used in a “semilog x” graph. The x domain will go from 0.001 up to 100 with ticks at 0.001, 0.01, 0.1, 1.0, 10 and 100. Conversely, the y domain will go from $-\pi$ up to $+\pi$, with linearly spaced ticks. The ticks will be inside the graph.

```

\fxsetnewstyle{my style A}{
  semilog x ,
  x ticks = {
    min = 0.001 ,
    max = 100 ,
    N = 6 ,
    units = rad/s ,
  } ,
  y ticks = {
    min = -3.14159265 ,
    max = 3.14159265 ,
    N = 8 ,
    units = rad ,
  } ,
  % the following key comes from pgfplot
  tick align=inside , %this package's default is outside.
}

```

Note: The *linear*, *loglog*, *semilog x* and *semilog y* keys are only used here for setting the ticks (linear or logarithmic). If none is given, it is assumed that both *x* and *y* axis are linear.

Note: If either *min* or *max* are missing, no tick list will be generated.

Note: In case of a logarithmic axis, both *min* and *max* must be greater than zero, otherwise an error will be raised.

4.3 Drawing a $f_i(x)$ Graph

There is a single drawing command `\fxgraphdraw` and a companion environment `fxgraph`, both share the same interface. The graph will be constructed as follow: 1. An outer `tikzpicture` environment 2. An inner `axis` environment 3. The function's graphs themselves 4. (in case of the `fxgraph` environment) further `pgfplot` commands. The `axis` environment will first be "styled" (as per *linear*, *loglog*, *semilog x* or *semilog y*, see 5) then the *ticks*, if defined, will be applied, lastly any further `pgfplot` key used when calling those commands.

Note: Given the above construct, generic `pgfplot` keys used will always have a precedence over the default styles, regardless of their order of appearance.

`\fxgraphdraw` `\fxgraphdraw {⟨keyval-list⟩}`

Creates a graph, and draw one or more functions/sets of functions as described by `⟨keyval-list⟩` (see below).

`fxgraph` `\begin{fxgraph} {⟨keyval-list⟩}`
`⟨further commands⟩`
`\end{fxgraph}`

Same as `\fxgraphdraw`, allowing to add further `pgfplot` and `tikz` commands.

Valid Keys when describing a graph:

<code>linear</code>	Both <i>x</i> and <i>y</i> axis are linear.
<code>loglog</code>	Both <i>x</i> and <i>y</i> axis are logarithmic.
<code>semilog x</code>	The <i>x</i> axis is logarithmic, the <i>y</i> is linear.
<code>semilog y</code>	The <i>y</i> axis is logarithmic, the <i>x</i> is linear.
<code>x ticks</code>	Describes the minor ticks to be drawn in the <i>x</i> axis. See below.
<code>y ticks</code>	Describes the minor ticks to be drawn in the <i>y</i> axis. See below.
<code>sans tikzpicture</code>	Suppress the outer <code>tikzpicture</code> environment.
<code>without tikzpicture</code>	Suppress the outer <code>tikzpicture</code> environment.
<code>function</code>	Adds a function specification, see 4.1.
<code>fx set</code>	A comma separated list of <code>fxsets</code> .
<code>use file</code>	(optional) This is equivalent of setting <code>use file</code> for every function to be drawn in this group, as in 4.1, the same conditions and warnings apply: The corresponding table files must exist but they won't be verified if they are up-to-date.

Besides those, any other `pgfplots` valid key can be used. (e.g. `red` , `thick` : the specific style will set the lines to be red and thick).

The `x ticks` and `y ticks` are set the same way as in 4.2 (`x tick=<keyval-list>`).

The `function` key defines (as in 4.1) a function to be draw, it can be used multiple times. Note that those functions will be drawn before any `fx set`.

`fx set` is a comma separated list of `<fxset>` (as defined in 4.1). All functions $f_i(x)$ associated with each `<fxset>` will be drawn.

Normally, the `\fxgraphdraw` command (viz-à-viz `fxgraph` environment) will insert an axis environment inside a `tikzpicture` environment. The `sans tikzpicture` and `without tikzpicture` keys will suppress that external `tikzpicture` environment.

Note: The `fxgraph` environment (as well the `\fxgraphdraw` command) usually extracts the many function's domain from `x ticks`, if present. Though, the domain can be set by other means (e.g. via `\fxsetnewstyle` or directly via `pgfplots` keys), a warning will be raised if `x ticks` isn't explicitly set. This warning can be suppressed with the package option: `no domain warning`, see 3.

Note: If the package option `use file` (see 3) is set, the local command/environment option `use file=false` can be used to locally revert it's effect. Note though, the function definition (see 4.1 has precedence over them).

5 Style's Default

For each kind of graph (linear, loglog, semilog) there is a corresponding predefined style:

`linear axis` This is the “base” style. Both axis will be gridded (lines at the corresponding ticks), with an axis line at the bottom (for x) and the left (for y). Ticks marks will be outside the graph. The legend (if present) will be at the top right of the graph. The $f_i(x)$ curves will be styled according to two lists: `fxgraph color list` and `fxgraph line list` (see below). The graph width is set to 80% of `\textwidth` and it's height to 35% of `\textwidth`.

`loglog axis` It applies the current `linear axis` style and the log basis is set to 10.

`semilog x axis` It applies the current `linear axis` style and the log basis is set to 10.

`semilog y axis` It applies the current `linear axis` style and the log basis is set to 10.

Those styles can be modified with `\pgfplotsset` or `\pgfkeys` (using `.style append` sub-key, for instance. See [2] and [1]). If using the `\pgfkeys` remember to switch first to the `pgfplots` ‘family’.

When styling a set of functions, two lists are used (`cycle multiindex* list` from `pgfplots`):

`fxgraph color list` Function's color will cycle through `red!80!black`, `green!80!black`, `blue!80!black`, `black`, `brown!70!black`, `teal!80!black`, `orange!80!black`, `violet!80!black`, `cyan!80!black`, `magenta!80!black`, `yellow!75!black` and `black!60!white`.

`fxgraph line list` Function's line style will cycle through `solid`, `solid`, `solid`, `dashed`, `dashed`, `dashed`, `dashdotdotted`, `dashdotdotted` and `dashdotdotted`.

Both can be redefined with `\pgfplotscreateplotcyclelist` from `pgfplots`[1].

6 Examples

6.1 Drawing a Bode Diagram

```
\fxsetnewstyle{db style}{
  semilog x ,
  y ticks = { min = -20 , max = 80 , N = 5 , units = db } ,
}

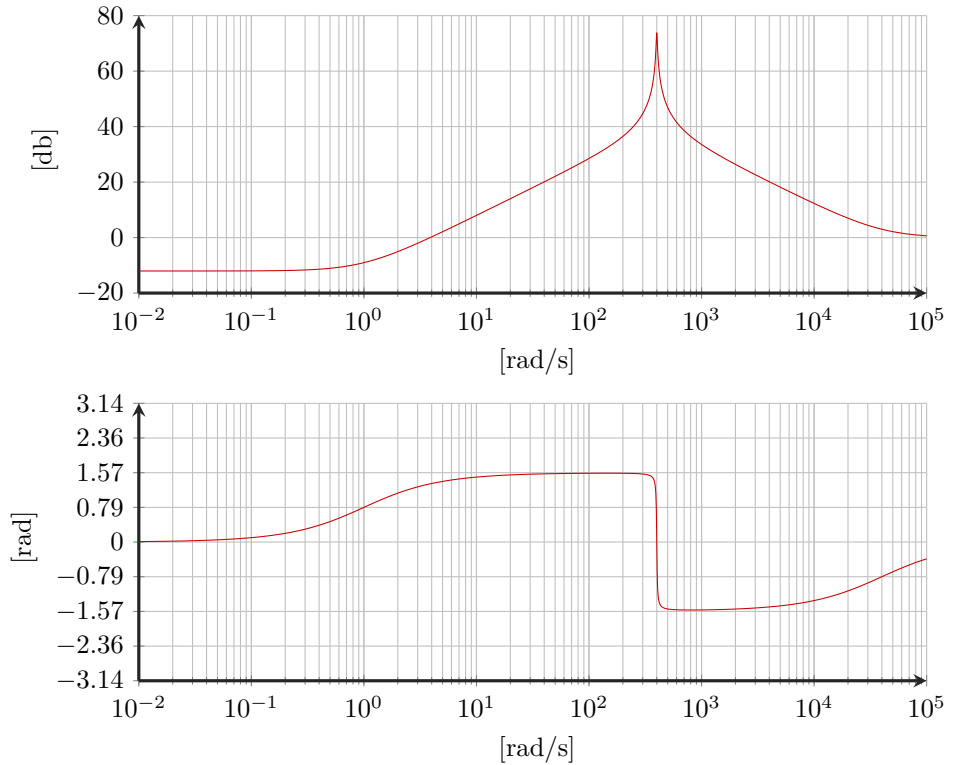
\fxsetnewstyle{phi style}{
  semilog x ,
  y ticks = { min = -3.14159265 , max = 3.14159265 , N = 8 , units = rad } ,
}

\fxsetnewstyle{freq range A}{
  semilog x ,
  x ticks = { min = 0.01 , max = 100000 , N = 7 , units = rad/s } ,
}

%
%% This is optional, just defining an auxiliary macro with f(x) core expression
%% Note that this is a valid gnuplot's expression (not LaTeX/TeX/...)
\def\Hs{(x*{0,1}+1)*(x*{0,1}+40000)/((-x^2+0.01*2*400*x*{0,1}+400^2))}

~\hfill
\fxgraphdraw{
  semilog x ,
  db style ,
  freq range A ,
  function={fx=20*log10(abs(\Hs))}
}

~\hfill
\fxgraphdraw{
  semilog x ,
  phi style ,
  freq range A ,
  function={fx={atan2( imag(\Hs) , real(\Hs) )}}
}
```



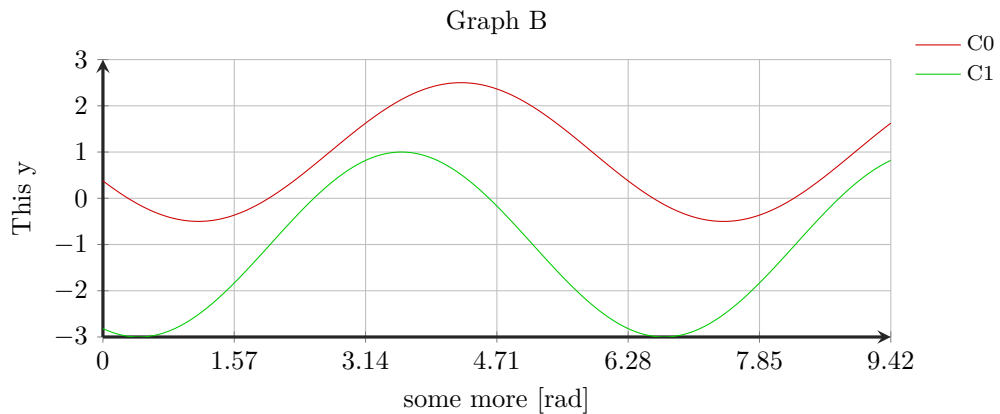
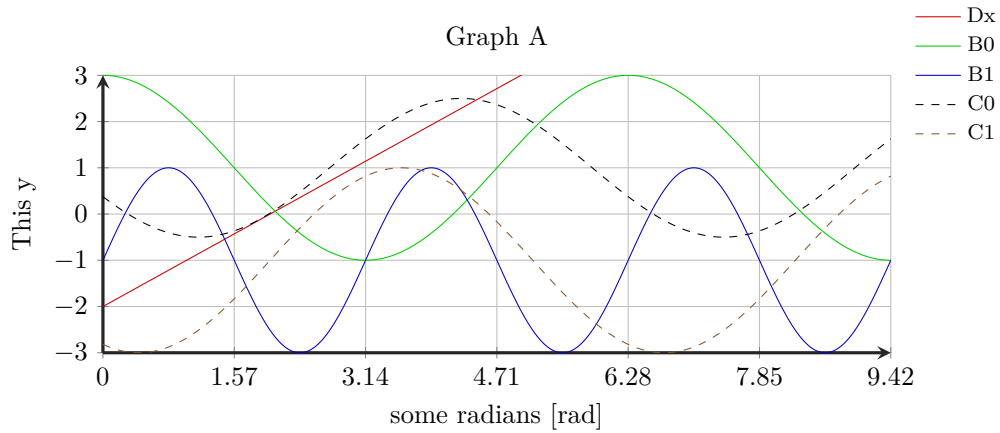
6.2 A Few Curves at Once

In the example below note that the *fx set* functions are drawn after the ‘Dx’. And that defines the legend order.

```
\fxsetnew{set-B}
\fxsetappend{set-B}{id=B0,fx=2*cos(x)+1,legend=B0}
\fxsetappend{set-B}{id=B1,fx=2*sin(2*x)-1,legend=B1,thick}
%the thick (line) keyword comes from tikz
\fxsetnew{set-C}
\fxsetappend{set-C}{id=C0,fx=1.5*cos(x+2)+1,legend=C0}
\fxsetappend{set-C}{id=C1,fx=2*sin(x-2)-1,legend=C1}

\fxgraphdraw{
  linear ,
  y ticks = {min = -3 , max = 3 , N = 6} ,
  x ticks = {min = 0 , max = 3*3.14159265 , N = 6 , units = rad} ,
  fx set = {set-B , set-C} ,
  function = {id=Dx,fx=x-2,legend=Dx} ,
  xlabel = some radians , % from pgfplots
  ylabel = This y , % from pgfplots
  title = Graph A % from pgfplots
}

\fxgraphdraw{
  linear ,
  y ticks = {min = -3 , max = 3 , N = 6} ,
  x ticks = {min = 0 , max = 3*3.14159265 , N = 6 , units = rad} ,
  fx set = {set-C} ,
  xlabel = some more , % from pgfplots
  ylabel = This y , % from pgfplots
  title = Graph B % from pgfplots
}
```

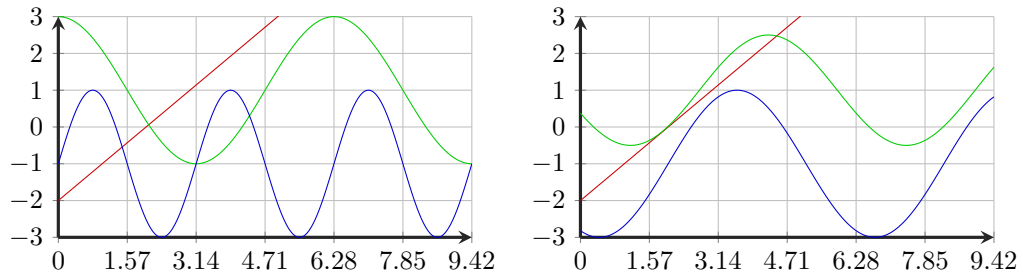


6.3 Further Customization

There are many ways, for instance, to have side by side graphs. One could use, for example, a *tabular* environment. In the following the *tikz* library *matrix* will be used, in which case the option *sans tikzpicture* is needed (besides a custom width and height for each graph). Note: Since *tikz* matrixes cannot be nested, and *pgfplots* legends are created as a *tikz* matrix, one can't have a legend in this case.

```
\fxsetnew{set-D}
\fxsetappend{set-D}{id=B0,fx=2*cos(x)+1}
\fxsetappend{set-D}{id=B1,fx=2*sin(2*x)-1,thick}
%the thick (line) keyword comes from tikz
\fxsetnew{set-E}
\fxsetappend{set-E}{id=C0,fx=1.5*cos(x+2)+1}
\fxsetappend{set-E}{id=C1,fx=2*sin(x-2)-1}

\begin{tikzpicture}
\matrix{
\fxgraphdraw[
  linear ,
  y ticks = {min = -3 , max = 3 , N = 6} ,
  x ticks = {min = 0 , max = 3*3.14159265 , N = 6} ,
  fx set = {set-D} ,
  function = {id=Fx,fx=x-2} ,
  sans tikzpicture ,
  width=0.47\textwidth , % from pgfplots
  height=0.30\textwidth , % from pgfplots
} &
\fxgraphdraw[
  linear ,
  y ticks = {min = -3 , max = 3 , N = 6} ,
  x ticks = {min = 0 , max = 3*3.14159265 , N = 6} ,
  fx set = {set-E} ,
  function = {id=Gx,fx=x-2} ,
  sans tikzpicture ,
  width=0.47\textwidth , % from pgfplots
  height=0.30\textwidth , % from pgfplots
} \\
};
\end{tikzpicture}
```



References

- [1] Christian Feuersänger. *The PGFPLOTS Package*. 2021, p. 573. URL: <http://mirrors.ctan.org/graphics/pgf/contrib/pgfplots/doc/pgfplots.pdf> (visited on 03/10/2025).
- [2] Till Tantau, Mark Wibrow, and Christian Feuersänger. *The TikZ and PGF Packages*. Institut für Theoretische Informatik / Universität zu Lübeck. 2023, p. 1321. URL: <http://mirrors.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf> (visited on 03/10/2025).
- [3] Thomas Williams and Colin Kelley. *gnuplot 5.4*. 2022, p. 316. URL: https://gnuplot.sourceforge.net/docs_5.4/Gnuplot_5_4.pdf (visited on 03/10/2025).