LaTeXiPy Example

Jean Nassar

August 22, 2017

Contents

List of Listings

List of Figures

1 Online resources

The Github repository is at https://github.com/masasin/latexipy, and the full example file is at examples/examples.py. Full documentation is available at https://latexipy.readthedocs.io/.

2 Assumptions

In order to generate plots with LaTeXiPy, install the package and import it.

This document assumes that the following imports are made:

```
from functools import partial
import matplotlib.pyplot as plt
import latexipy as lp
```

Listing 1: The imports used in this example.

Also, it assumes that there is a function, `plot_sin_and_cos()` which uses a `matplotlib`-based package to generate a plot without calling `plt.savefig()` or `plt.close()`. LaTeXiPy is known to work well with various libraries, including `matplotlib`, `numpy`, `pandas` and `seaborn`, among others.

3 Plotting

3.1 Without LaTeXify

If you don't LaTeXify, `matplotlib`'s defaults are used. The typeface is sans-serif, and the font a bit larger. The code in Listing 2 generates Figure 1.

```
with lp.figure('sincos'):
    plot_sin_and_cos()
```

Listing 2: Generate figures with one extra line.

`figure` is a context manager with the following signature:

```
@contextmanager
def figure(filename, *, directory='img', exts=['pgf', 'png'],
           size=None, mkdir=True):
```

Listing 3: The signature for `lp.figure()`.

Note that the default for `directory`, `./img`, is relative to the calling location, and that `size` is a tuple with the x- and y-dimensions in inches.

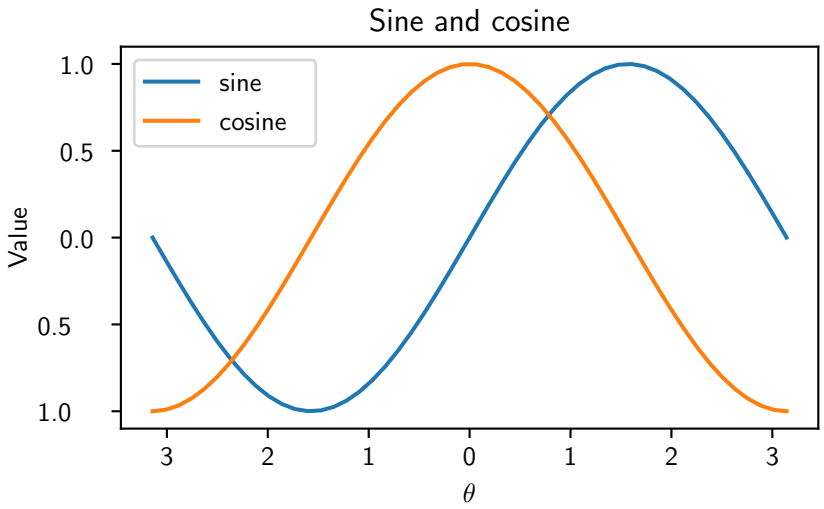

Figure 1: The plot, with default parameters before LaTeXifying.

3.2 With LaTeXify

Figure 2 shows a plot using the default LaTeXiPy configuration, with the same typeface as the body, but a size of 8 pt. To generate it, we can use the code in Listing 4.

```
lp.latexify()

with lp.figure('sincos'):
    plot_sin_and_cos()
```

Listing 4: `lp.latexify()` generates plots that fit well with LaTeX.

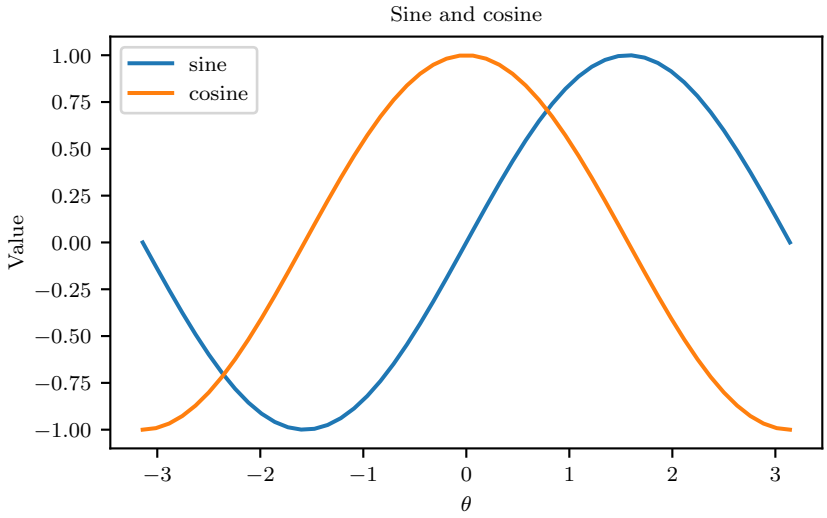

Figure 2: The plot, with default parameters after LaTeXifying.

3.3 Custom parameters

`lp.latexify()` uses `lp.PARAMS` by default. Its values are shown in Listing 5.

```
FONT_SIZE = 8

PARAMS = {
    'pgf.texsystem': 'xelatex',  # pdflatex, xelatex, lualatex
    'text.usetex': True,
    'font.family': 'serif',
    'font.serif': [],
    'font.sans-serif': [],
    'font.monospace': [],
    'pgf.preamble': [
      r'\usepackage[utf8x]{inputenc}',
      r'\usepackage[T1]{fontenc}',
      ],
    'font.size': FONT_SIZE,
    'axes.labelsize': FONT_SIZE,
    'axes.titlesize': FONT_SIZE,
    'legend.fontsize': FONT_SIZE,
    'xtick.labelsize': FONT_SIZE,
    'ytick.labelsize': FONT_SIZE,
}
```

Listing 5: The default parameters changed by LaTeXify.

To change some parameters, such as the font size, all you need to do is pass a different dictionary to `lp.latexify()`. This can be done at any time. Listing 6 shows an example of increasing the font size.

```
    plot_sin_and_cos()

# You can change the parameters at any time. To increase the font size:
font_size = 10
params = lp.PARAMS.copy()
params.update({param: font_size
```

Listing 6: Increasing the font size is as simple as changing the default values.

After the parameters have been updated, running Listing 2 again gives Figure 3, with a 10 pt font.

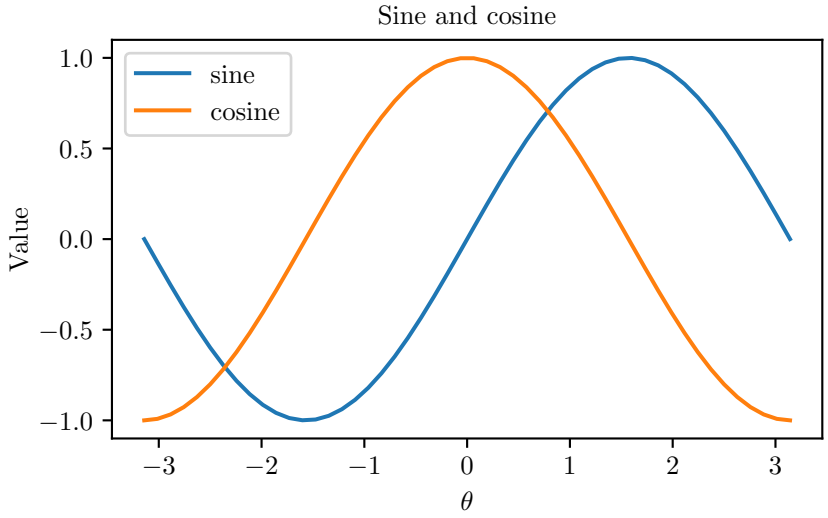

Figure 3: The plot, with the font size increased.

4 Programming tips

If you keep passing the same arguments to `lp.figure()` (for example, an output directory, a set of filetypes, or a certain size), you can save it for reuse by using the `partial` function from the `functools` module, as shown in Listing 7. After that, you can use it just like `lp.figure()`. Note that you would not be able to redifine an argument that you had previously applied.

Here, `DIRECTORY` refers to any output directory, either as a string, or as a `pathlib.Path` object.

```
figure = partial(lp.figure, directory=DIRECTORY)

with figure('sincos_partial'):
    plot_sin_and_cos()
```

Listing 7: The `sincos_partial` plots will be stored in `DIRECTORY`.

5 Using in LaTeX

To include a PGF file in your LaTeXdocument, make sure that the `pgf` package is loaded in the preamble:

```
\usepackage{pgf}
```

After that, you can include it in the correct location with:

```
\input{<filename>.pgf}
```

Listing 8 shows a minimum working example of adding an image within a figure.

```
\documentclass{article}

\usepackage{pgf}

\begin{document}
  \begin{figure}[h]
    \centering
    \input{img/filename.pgf}
    \caption[LOF caption]{Regular caption.}
    \label{fig:pgf_example}
  \end{figure}
\end{document}
```

Listing 8: A minimum working example of using PGF with LaTeX.

Note that figures using additional raster images can only be included by `\input` if they are in the same directory as the main LaTeXfile. To load figures from other directories, you can use the `\import` package instead.

```
\usepackage{import}
\import{<path to file>}{<filename>.pgf}
```

An example is given in Listing 9.

```
\documentclass{article}

\usepackage{import}
\usepackage{pgf}

\begin{document}
  \begin{figure}[h]
    \centering
    \import{/path/to/file/}{filename.pgf}  % Note trailing slash.
    \caption[LOF caption]{Regular caption.}
    \label{fig:pgf_example}
  \end{figure}
\end{document}
```

Listing 9: Importing a raster-using PGF from a different directory requires the `\import` package.