The PracTEX Journal, 2010, No. 1
Article revision 2010/05/25

Useful Vector Graphic Tools for LATEX Users

T. Morales de Luna

Email Tomas.Morales@uco.es
Address Dpto. de Matemáticas
Escuela Politécnica Superior
Universidad de Córdoba
Córdoba 14071
Spain

Abstract This paper presents some useful tools for creating vector graphics that can be included in LATEX documents. Of all the tools available, we focus on those that can produce graphics easily, and that can include any LATEX math formula. In particular, we present three useful tools: `Xfig`, `LaTeXDraw`, and `Matplotlib`. While the two first are intended to produce sketches and figures, the last will produce graphs, charts and contours.

1 Introduction

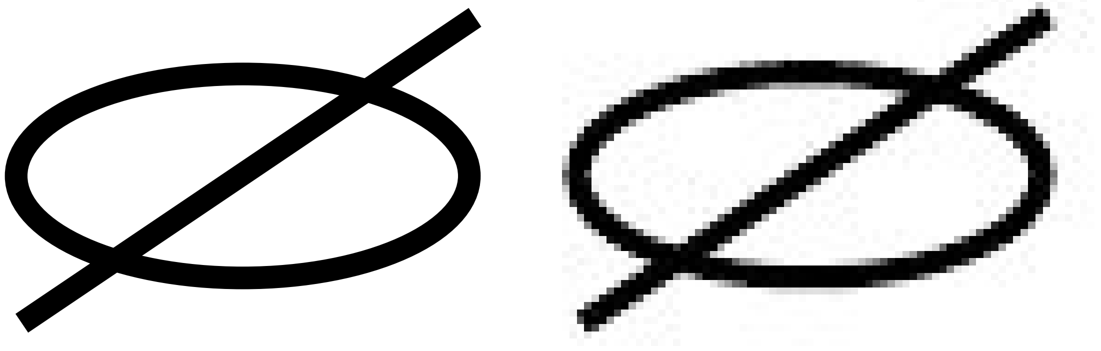
In this article, we describe some vector graphic tools that work with LATEX code so that users can easily produce *good* graphics to be included in LATEX documents.

2 Including graphics in LATEX documents

Although this article focuses on tools for creating graphics rather than on LATEX packages for including graphics, it is important to say a few words about how to include graphics in `tex` documents. For more detail, please see [1] and [2].

First of all, it is important to recall the difference between vector and bitmap graphics. While vector graphics behave well for scaling and rotation without loss of quality, the same is not true with bitmap graphics. So, whenever possible, we will use vector graphics in our documents.

Once you have drawn your figures, you can easily include them in your document by using the package `graphicx`. To use this package, include the following in the preamble of your `tex` document:



(a) Vector graphic (b) Bitmap graphic

Figure 1: Difference between vector and bitmap graphics.

```
\usepackage{graphicx}
```

or if you are going to produce a `pdf` file with `pdflatex`, use

```
\usepackage[pdftex]{graphicx}
```

Once this package is in place you can include a figure by writing

```
\includegraphics[options]{myfigure}
\caption{mycaption}
```

For example, Figure 1 presents two subfigures placed side by side. This was made possible by using

```
\usepackage{subfigure}
```

. This shows how it is used:

```
\begin{figure}
  \centering
  \subfigure[Vector graphic]
  {
    \includegraphics[width=0.47\linewidth]{vector}
  }
  \subfigure[Bitmap graphic]
  {
    \includegraphics[width=0.47\linewidth]{bitmap}
  }
\caption{Difference between vector and bitmap graphics}
\end{figure}
```

An interesting feature of the `graphicx` package for figures is that it lets you scale, rotate, trim, and apply other features. For further details, please refer to [1] and [3].

Another interesting TEX macro package for generating graphics is `pgf`. It is platform- and format-independent and works together with the most important TEX back-end drivers, including `pdftex` and `dvips`. It comes with a user-friendly syntax layer called `TikZ`. See [2] for more details.

3 Creating graphics

A common question is, *How do I include formulas or TEX code in the picture?*. Well, you can use your favorite bitmap graphic editor and, when possible, one that produces vector graphics, but it is possible your favorite graphics editor may not be able to do the job. A work-around might be to generate two pictures: a formula using a temporary TEX file, and the background picture, drawn in your bitmap editor. Then you copy the formula and paste it on the background picture. We know that this is not a good practice because of the low-quality image, unless you do it with care (meaning that you should maintain the vector properties of the image when copying-pasting).

Another option is to generate just the picture, include it in your document and then place the formulas over it by including the command `\pgfputat`. This command lets you place almost anything at a given absolute position. For instance, the code below will produce Figure 2.

```
\begin{pgfpicture}{0cm}{0cm}{3cm}{3cm}
\pgfputat{\pgfxy(1.5,0)}{\pgfbox[center,center]{$\int_0^\infty x dx$}}
\includegraphics[width=3cm]{vector}
\end{pgfpicture}
```

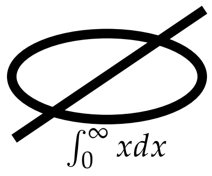


Figure 2: Using the command `\pgfputat`

The problem is to put the formulas at the correct position, but you will discover techniques that give the desired result.

Another method is to generate the picture using `Pstricks`. This way, you have control over your picture. Consider the example given in the `pgf` user guide that produces the Figure 3:

```
\begin{pgfpicture}{0cm}{0cm}{5cm}{2cm}
\pgfputat{\pgfxy(1,1)}{\pgfbox[center,center]{Hi!}}
\pgfcircle[stroke]{\pgfxy(1,1)}{0.5cm}
\pgfsetendarrow{\pgfarrowto}
\pgfline{\pgfxy(1.5,1)}{\pgfxy(2.2,1)}
\pgfputat{\pgfxy(3,1)}{
\begin{pgfrotateby}{\pgfdegree{30}}
\pgfbox[center,center]{$\int_0^\infty xdx$}
\end{pgfrotateby}}
\pgfcircle[stroke]{\pgfxy(3,1)}{0.75cm}
\end{pgfpicture}
```

This can be tricky and in general it is more difficult to draw graphics with commands than by using your mouse.

3.1 The `Xfig` program

`Xfig` [4] is a program that can draw vector graphics, and easily combine them with LaTeX formulas. Although this program has a challenging graphical user interface, it is still a handy tool.

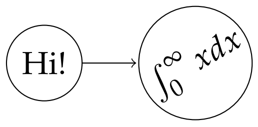


Figure 3: This example was captured from `pgf` user guide.

In order to include LaTeX code with graphics, we have to launch the program with the *special-text* flag.

➤ Step 1. Run the following command in a shell

```
xfig -specialtext
```

Once `xfig` is running, you can draw pictures and place any LaTeX equation or formula with the *insert text tool*. Use this in the usual way, but put the LaTeX markup between $ symbols. When finished, save your picture.

➤ Step 2. Draw your picture and add formulas between $ symbols.

➤ Step 3. Save the obtained figure with `.fig` extension.

Next, we are going to use the shell command `fig2dev` to produce the desired figure. Here, I assume that you used the filename `myfigure.fig`.

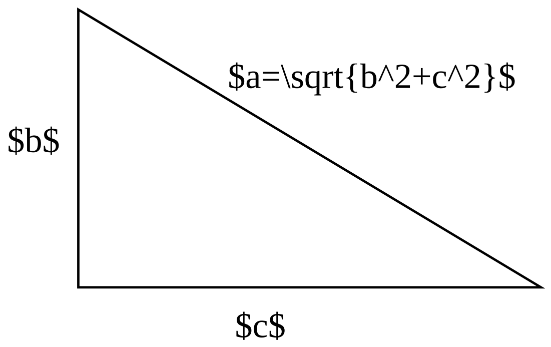
➤ Step 4. Run the following commands in the shell.

```
fig2dev -L pstex myfigure.fig > myfigure.pstex_t
fig2dev -L pstex_t -p myfigure.pstex_t myfigure.fig > myfigure.temptex
```

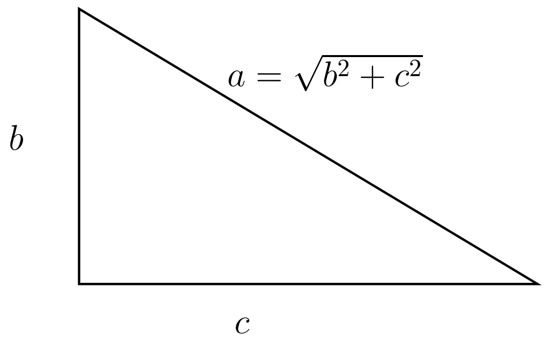
The first command generates `.ps` from `.fig`, and the second one generates `.tex` commands from `.fig` based on the specifications in the `.ps` file.

➤ Step 5. Create the file `myfigure.tex` with the following content

```
\documentclass{article}
\usepackage{graphicx,epsfig,color}
\pagestyle{empty}
\begin{document}
\input{myfigure.temptex}
\end{document}
```



(a) Plot produced with Xfig



(b) Final figure obtained

Figure 4: Using `Xfig` to produce graphics

➤ Step 6. Create the corresponding `.eps` file by running the following commands in a shell

```
latex myfigure.tex
dvips -E -q -o myfigure.eps myfigure.dvi
```

If you prefer a `.pdf` file, use `epstopdf` to convert the `.eps` figure to `.pdf`.
You may put all the previous commands in a script, or you may use the one available in [5].

3.2 The `LatexDraw` program

A more recent program for drawing graphics is `LaTeXDraw`[6]. It is a free `PSTricks` code generator or `PSTricks` editor distributed under the GNU GPL. `LaTeXDraw` was developed in `JAVA`, so it runs in any operating system. The graphical user interface of `LaTeXDraw` is similar to the ones found in many graphics editors.

➤ Step 1. Draw your picture and include formulas between $
Since the resulting picture will be used in a LaTeX document, you may place a formula using the text button available in the toolbar in `LaTeXDraw`. Type the formulas between $ as you would normally do.

➤ Step 2. Export the picture as `PSTricks` in a `tex` file.
Once you have drawn your picture, you can save the `PSTricks` code with the menu File → Export as → PSTricks code.

A figure similar to 4(a) will produce the following `.tex` file with `LaTeXDraw`:

```
% Generated with LaTeXDraw 2.0.3
% Tue Dec 29 12:38:16 CET 2009
% \usepackage[usenames,dvipsnames]{pstricks}
% \usepackage{epsfig}
% \usepackage{pst-grad} % For gradients
% \usepackage{pst-plot} % For axes
\scalebox{1} % Change this value to rescale the drawing.
{
\begin{pspicture}(0,-1.5507812)(6.1871877,1.5307813)
\pspolygon[linewidth=0.04](0.701875,1.5107813)(0.701875,-0.94921875)
(5.481875,-0.96921873)
\usefont{T1}{ptm}{m}{n}
\rput(0.25546876,0.22578125){$b$}
\usefont{T1}{ptm}{m}{n}
\rput(2.5754688,-1.3742187){$c$}
\usefont{T1}{ptm}{m}{n}
\rput(4.3554688,0.8857812){$a=\sqrt{b^2+c^2}$}
\end{pspicture}
}
```

Be sure to include the corresponding packages if you use this code in your document.

If you prefer to generate independent vector graphics that you can include in your document with the `\includegraphics` command, you can do so by creating the file `myfigure.tex` as follows.

➤ Step 3. Create a new `.tex` with the following content.

```
\documentclass{article}
\usepackage{pstricks}
\usepackage{pst-plot}
\usepackage{pst-eps}
\usepackage{pst-grad}
\pagestyle{empty}
\begin{document}
```

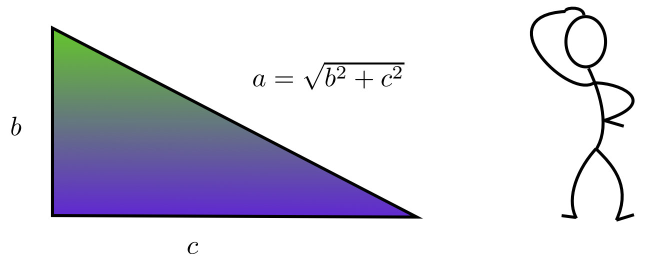


Figure 5: A figure drawn with `LaTeXDraw`.

```
\begin{TeXtoEPS}
\begin{pspicture}
<...>
\end{pspicture}
\end{TeXtoEPS}
\end{document}
```

Replace <. . . > with the corresponding code generated by `LaTeXDraw`. It is important to place the `pspicture` inside the environment `TeXtoEPS`, as this will provide LaTeX with the correct bounding box for the picture. If you omit it, you may get a final picture that has been trimmed or in a page size far larger than the size of the picture. You must not include all the packages that are shown above, but just the ones needed for your graphic. Now, all that remains to do is:

➤ Step 4. Execute the commands

```
latex myfigure.tex
dvips -E -q -o myfigure.eps myfigure.dvi
```

With this program you can easily draw professional-looking graphics that include LaTeX formulas, like the one shown in 5.

4 `Matplotlib` in LaTeX documents

When writing technical documents with LaTeX, it is common to plot graphs, charts, contour plots etc using `Matlab`, `Mathematica`, or other software, and then include them in your document.

On the one hand, `Matlab` and `Mathematica` are good commercial programs, but there are other free alternatives such as `Matplotlib` [7]. `Matplotlib` is a plotting library for the `Python` programming language and its `NumPy` numerical mathematics extension. It provides an object-oriented API which allows you plot picturess to be embedded into applications using generic GUI toolkits, like `wxPython`, `Qt`, or `GTK`. There is also a procedural `pylab` interface based on a state machine (like `OpenGL`), designed to closely resemble `Matlab`. The advantage is that it is open-source software.

You could use `Matplotlib` to generate your plots and then include them in your documents, but you can also include a LaTeX package that lets you to use `Python` code directly in your document: `python` [8]. To do this, add the line `\usepackage{python}` to the preamble and then insert any `Python` code inside the environment `python` (you may need to download this package if you do not have it in your system). In this way, the images are automatically generated when you compile your `.tex` by including the flag `-shell-escape` to the command line.

To use this method, do the following:

- ➤ Step 1. Write `\usepackage{python}` before the `\begin{document}`
- ➤ Step 2. Insert any `Python` code inside the environment `python`
- ➤ Step 3. Compile your `tex` file using the command

```
pdflatex -shell-escape your_tex_file.tex
```

Let's look at some examples.

By writing the following code into the file `example.tex`

```
\documentclass{article}
\usepackage[pdftex]{graphicx}
\usepackage{python}
\begin{document}
\begin{figure}
{\Huge Document including a plot}
\begin{python}
from pylab import *
x = linspace(-4*pi,4*pi,200)
plot(x,sin(x)/x)
```

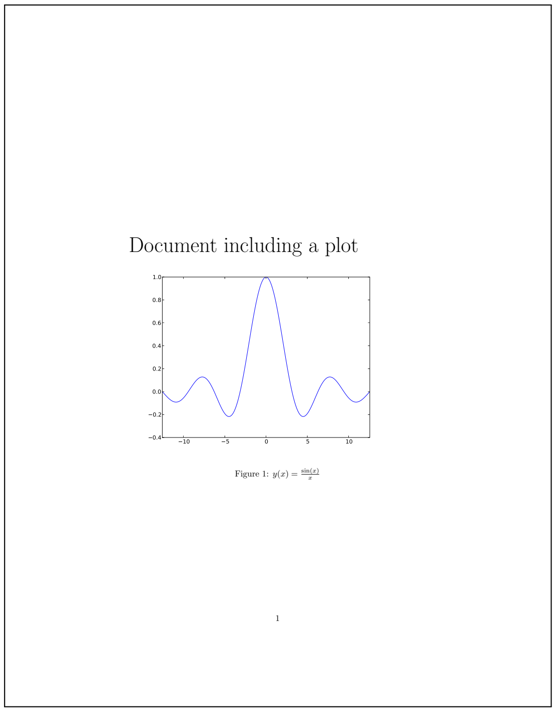

Document including a plot

Figure 1: $y(x) = \frac{\sin(x)}{x}$

1

Figure 6: Plotting $y = \dfrac{\sin(x)}{x}$ with python

```
xlim(-4*pi,4*pi)
savefig('plot-include.pdf')
print r'\includegraphics[width=0.9\linewidth]{plot-include.pdf}'
\end{python}
\caption{$y(x)=\frac{\sin(x)}{x}$}
\end{figure}
\end{document}
```

and then executing the command

```
pdflatex -shell-escape example.tex
```

we get the pdf file corresponding to Figure 6.

There are many other examples. For instance, the code

```
\begin{python}
from pylab import *
x = arange(-3.0, 3.0, .025)
y = arange(-3.0, 3.0, .025)
X, Y = meshgrid(x, y)
Z1 = bivariate_normal(X,Y,1.0,1.0,0.0,0.0)
Z2 = bivariate_normal(X,Y,1.5,0.5,1,1)
Z = 10.0 * (Z2 - Z1)
CS = contour(X, Y, Z)
clabel(CS, inline=1, fontsize=10)
savefig('plot-include2.pdf')
print r'\includegraphics[width=0.9\linewidth]{plot-include2.pdf}'
\end{python}
```

will produce Figure 7(a), and

```
\begin{python}
from pylab import *
figure(1, figsize=(6,6))
ax = axes([0.1, 0.1, 0.8, 0.8])
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
fracs = [15,30,45, 10]
explode=(0, 0.05, 0, 0)
pie(fracs, explode=explode, labels=labels, autopct='%1.1f%%', shadow=True)
title('Raining Hogs and Dogs', bbox={'facecolor':'0.8', 'pad':5})
savefig('plot-include3.pdf')
print r'\includegraphics[width=0.9\linewidth]{plot-include3.pdf}'
\end{python}
```

will produce Figure 7(b).

`Matplotlib` can even include LaTeX commands as is shown in [9].

5 Conclusion

We have shown several ways to include graphics in LaTeX documents using two programs: `Xfig` and `LaTeXDraw`. While both give good results, the second is more recent and user-friendly.

Both of these programs will draw sketches and similar figures, but in order to do graphs, charts or contour plots, `Matplotlib` is the better choice. You can also embed `Matplotlib` and `Python` code directly in your document.

References

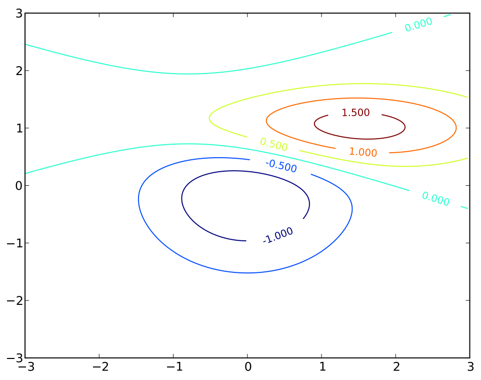
[1] Klaus Hoeppner. *Strategies for including graphics in LaTeX documents*. The PracTEX Journal, 2005. No 3.
`http://www.tug.org/pracjourn/2005-3/hoeppner/`

[2] Claudio Beccari . *Graphics in LaTeX*. The PracTEX Journal, 2007. No 1.
`http://www.tug.org/pracjourn/2005-3/hoeppner/`

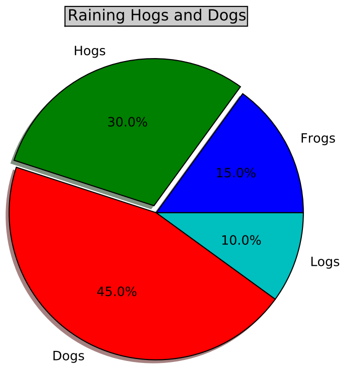
[3] *Enhanced support for graphics*
`http://www.ctan.org/tex-archive/macros/latex/required/graphics/`

[4] *Xfig*.
`http://www.xfig.org/`

[5] *Conversion tools*.
`http://www.few.vu.nl/~wkager/tools.htm`



(a)



(b)

Figure 7: Using `Matplotlib` to produce graphics

[6] *LaTeXDraw*.
http://latexdraw.sourceforge.net

[7] *Matplotlib*
http://matplotlib.sourceforge.net/index.html

[8] *Embedding python into LaTeX*
http://www.imada.sdu.dk/~ehmsen/pythonlatex.php

[9] *pylab_examples example code: integral_demo.py*.
http://matplotlib.sourceforge.net/examples/pylab_examples/integral_demo.html